Les grammaires d'interaction

Guy Perrier

NANCY 2 UNIVERSITE

IUT D'Epinal, département OGP — École doctorale IAE + M

Département de formation doctorale en informatique

Les grammaires d'interaction

MÉMOIRE

12 novembre 2003

pour l'obtention de l'

Habilitation de l'Université Nancy 2

(Spécialité Informatique)

par

Guy Perrier

Composition du jury

Rapporteurs :	Anne Abeillé
	Alexandre Dikovsky
	Gérard Huet
Examinateurs :	Philippe Blache
	Patrick Blackburn
	Jeanine Souquières
	Mark Steedman

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier tous les membres du jury pour leurs remarques pertinentes qui m'ont permis d'enrichir ce mémoire et pour leurs encouragements à poursuivre ce travail de recherche.

Je voudrais aussi à remercier tout particulièrement Guillaume Bonfante et Bruno Guillaume qui ont participé à l'implémentation de l'analyseur syntaxique LEOPAR fondé sur les grammaires d'interaction. Cette implémentation a beaucoup contribué à la crédibilité du formalisme des grammaires d'interaction.

à mes parents,
à Mariola

Table des matières

introduction

Ce mémoire a pour objet de présenter un formalisme linguistique original, les grammaires d'interaction, qui se propose d'embrasser la syntaxe et la sémantique des langues.

A l'heure où la "linguistique de corpus" et les méthodes statistiques semblent avoir pris le pas sur les approches symboliques en linguistique informatique, n'est-ce pas un combat d'arrière garde que de proposer un nouveau formalisme linguistique ? Espérons que ce mémoire aidera à montrer qu'il n'en est rien car il est motivé par la conviction que les approches symboliques sont loin d'avoir été au bout de leurs possibilités dans la modélisation des langues et par là même dans leur contribution à la linguistique tout court. Directement en prise sur les connaissances linguistiques, elle cherchent d'abord à les mathématiser. Même si cette mathématisation aboutit à des modèles formels encore bien grossiers relativement à toute la subtilité et la richesse des langues, elle n'en est pas moins fort utile pour aller plus loin dans la connaissance de celles-ci [Cho57]. La modélisation mathématique joue ici un rôle analogue à celui qu'elle joue dans toute autre science expérimentale. Un modèle linguistique constitue d'une certaine façon une hypothèse qu'il s'agit ensuite de confronter à la réalité de la langue à travers l'expérimentation, la matière d'expérimentation étant les corpus et l'outil étant l'ordinateur et les logiciels de traitement de ces corpus.

Dans ce mémoire, nous ne prétendons pas présenter un modèle linguistique au sens d'une théorie linguistique formalisée mais seulement un cadre formel pour de tels modèles. Bien entendu, notre propos sera accompagné d'exemples, empruntés au français, mais ces exemples ne seront là que pour illustrer les différentes facettes du formalisme. Aller au delà exigerait des compétences linguistiques que nous ne possédons pas.

Bien entendu, un formalisme n'est pas non plus complètement neutre par rapport à toute théorie linguistique. Ainsi, nous reprenons à notre compte bien des motivations mises en avant par C. Pollard et I. Sag [PS94] quand ils ont conçu HPSG. Comme eux, nous avons fait le choix d'un formalisme déclaratif qui se présente comme un système de contraintes et qui est indépendant du processus de compréhension ou de génération de la langue. Comme eux, nous pensons que la possibilité d'exprimer la sous-spécification est essentielle. Une intégration souple de la sémantique nous semble aussi importante de même que la possibilité d'effectuer des analyses incrémentales et robustes. Enfin, il faut que linguistes et informaticiens puissent tous s'y retrouver : les premiers doivent pouvoir exprimer de façon simple et lisible dans le formalisme les généralités de langue qu'ils mettent en évidence ; les seconds doivent pouvoir concevoir des algorithmes permettant le traitement automatique dans des li-

mites acceptables.

Dans ce mémoire, nous laisserons totalement de côté l'utilisation des statistiques. D'une part, nous ne chercherons pas à effectuer une évaluation comparative de notre formalisme avec les approches basées sur les statistiques. Il faudrait pour cela disposer de grammaires d'interaction à large couverture qui permettent d'analyser des corpus réels. Comme nous le verrons à la fin du mémoire, la construction de telles grammaires est tout un chantier qui n'en est qu'à ses débuts. D'autre part, nous aurions pu intégrer des probabilités dans notre formalisme comme G. Bonfante et P. de Groote l'ont fait pour les grammaires catégorielles [BdG01] mais cela semblait prématuré aussi.

Plan du mémoire

1 - Des réseaux de démonstration aux grammaires d'interaction

Les grammaires d'interaction sont issues des grammaires catégorielles [Adj35, Moo96, Ret00]. L'idée fondamentale des grammaires catégorielles est que les syntagmes sont vus comme des ressources consommables positives ou négatives ; les ressources négatives représentent des syntagmes attendus et les ressources positives des syntagmes disponibles et l'analyse syntaxique peut se résumer à un processus de neutralisation dans lequel chacun cherche son complément.

Une façon classique de présenter les grammaires catégorielles est de le faire sous une forme logique à l'aide du calcul de Lambek [Lam58]. En combinant la sensibilité aux ressources et la non commutativité, cette logique se prête bien à la modélisation de la syntaxe des langues mais, en même temps, cette combinaison entraîne une rigidité qui limite grandement son pouvoir d'expression. Un moyen de relâcher cette rigidité consiste à enrichir le cœur logique à l'aide de modalités [Moo96].

L'approche logique n'est pas la seule possible et M. Steedman propose les grammaires catégorielles combinatoires (Combinatory Categorial Grammar ou CCG) [Ste00] qui est un calcul de combinateurs qui étend le pouvoir d'expression des grammaires catégorielles aux langages faiblement contextuels.

Nous avons choisi une voie qui nous a été inspiré par l'étude des réseaux de démonstration de la logique linéaire intuitionniste [Lam96]. Nous avons développé un premier formalisme fondé sur les réseaux de démonstration étiquetés [Per99]. Dans ce formalisme, un réseau de démonstration modélise les dépendances syntaxiques dans une phrase. Les étiquettes qui représentent la forme phonologique des syntagmes vont permettre d'exprimer de façon plus souple que dans les grammaires de Lambek deux aspects qui sont ici dissociés : la linéarité des dépendances entre syntagmes et les contraintes sur l'ordre de ces syntagmes.

Dans d'autres travaux [Per01], nous avons montré qu'une démonstration dans le fragment implicatif de la logique linéaire intuitionniste pouvait se réduire à la recherche de modèles particuliers de descriptions d'arbres. La notion de *description d'arbre* ne vient pas de la logique mais de la linguistique informatique. Elle a été introduite par M. Marcus, D. Hindle et M.Fleck pour réduire l'indéterminisme dans l'analyse syntaxique des langues [MHF83]. Elle a été

reprise par K. Vijay-Shanker pour modéliser l'opération d'adjonction des TAG sous une forme monotone [VS92]. Elle a été ensuite systématiquement étudiée d'un point de vue mathématique par J. Rogers et K. Vijay-Shanker[RVS92] et a donné lieu à divers formalismes issus des TAG [Kal99, RVSW01, Mus01] ou pas [Bla01].

Classiquement, une description d'arbre est un ensemble de nœuds et de relations de domination, de parenté et de préséance entre ces nœuds. L'intérêt de cette notion est qu'elle permet d'exprimer la sous-spécification d'arbres. Dans le domaine linguistique, les arbres concernés sont les arbres syntaxiques et dans le domaine de la logique linéaire intuitionniste, ils représentent l'ordre d'utilisation des formules logiques atomiques dans une démonstration. En début de démonstration, un tel arbre est sous-spécifié et les formules atomiques qui en constituent les nœuds sont polarisées pour exprimer le fait qu'elles sont des ressources disponibles ou attendues. La démonstration consiste à spécifier progressivement l'arbre de préséance entre formules en faisant se rencontrer une ressource disponible avec le besoin correspondant. C'est dans cette rencontre qu'apparaît le caractère linéaire de la logique puisque une ressource est exactement consommée par un besoin. La démonstration réussit lorsque l'on arrive à un arbre complètement spécifié et neutre c'est-à-dire où toutes les polarités se sont trouvées annulées.

Cette vision de la démonstration en logique linéaire intuitionniste a constitué le point de départ pour l'ébauche d'une première version des grammaires d'interaction que nous avons appelée les *grammaires d'interaction primitives*. Les objets de base y sont les descriptions d'arbres syntaxiques que nous appellerons dans le reste du mémoire *descriptions syntaxiques*. Dans ces descriptions, les nœuds représentent des syntagmes et les relations expriment les dépendances entre ces syntagmes. Ces nœuds sont polarisés : négatifs, ils représentent des syntagmes attendus et, positifs, des syntagmes disponibles. L'analyse syntaxique va consister à fusionner pas à pas les nœuds positifs avec des nœuds négatifs correspondants pour spécifier la description syntaxique et aboutir en cas de succès à un arbre syntaxique complètement spécifié où tous les nœuds ont été neutralisés.

2 - Le modèle syntaxique des grammaires d'interaction

Dans la plupart des formalismes syntaxiques basés sur les arbres, la composition syntaxique se réduit à de la composition d'arbres : on accroche les arbres les uns aux autres en identifiant la racine de l'un à une feuille d'un autre. Par exemple, l'adjonction des TAG peut être vue comme une double composition d'arbres qui devient un entrelacement dans le formalisme plus sophistiqué des DTSG (Description-Tree Substitution Grammars) [RVSW01].

L'originalité des grammaires d'interaction est que la composition syntaxique est guidée par un mécanisme basé sur la notion de polarités. Cette notion est utilisée dans plusieurs formalismes syntaxiques avec l'idée commune de besoins et ressources cherchant à se compléter mais avec des particularités quant à la façon de réaliser cette idée [Sta97, Mus01, DT99, Dik01]. Elle recoupe aussi l'idée de valence de Tesnière [Tes59]. Dans les grammaires d'interaction primitives, la composition syntaxique apparaît comme un processus électro-

statique dans lequel l'opération élémentaire consiste à identifier deux nœuds syntaxiques duaux. Cette neutralisation ne se passe pas seulement entre la racine d'un arbre et une feuille d'un autre si bien que cela permet de superposer partiellement des arbres syntaxiques, ce qui est fondamental du point de vue du pouvoir d'expression du formalisme.

A ce sujet, les grammaires d'interaction primitives sont néanmoins limitées par l'impossibilité de superposer des nœuds qui sont neutres. A l'opposé, elles ne sont pas suffisamment contraintes pour ce qui est des relations de parenté et de domination : on voudrait pouvoir dans une description clore l'ensemble des fils d'un nœud syntaxique et imposer des propriétés aux chaînes de nœuds qui réalisent les relations de domination. Enfin, le raffinement de la notion de polarité en l'attachant non plus aux nœuds mais aux traits morpho-syntaxiques décrivant ses nœuds permettrait une plus grande subtilité dans le pouvoir d'expression du formalisme.

C'est tout ceci qui est pris en compte dans le formalisme des *grammaires d'interaction* tout court. Les objets de base restent toujours les descriptions syntaxiques mais ce ne sont plus les nœuds de ces descriptions qui sont polarisés. Ce sont les traits morpho-syntaxiques attachés à ces nœuds. Alors que habituellement, un trait morpho-syntaxique est associé à des valeurs dans des couples *trait-valeur*, dans les grammaires d'interaction, un trait est associé à des valeurs et des polarités dans des triplets *trait-polarité-valeur*. Une polarité peut être –1, 0 ou +1 selon que l'on à faire à un trait négatif, neutre ou positif.

La composition syntaxique de deux descriptions d'arbre apparaît toujours comme un processus électrostatique mais l'opération élémentaire consiste maintenant à identifier deux nœuds syntaxiques porteurs de traits opposés. Nous appellerons cette opération *neutralisation de traits opposés*.

Les grammaires d'interaction sont lexicalisées. Un lexique associe à chaque mot de la langue un ensemble de descriptions syntaxiques élémentaires. Analyser une phrase, consiste tout d'abord à sélectionner pour chaque mot de la phrase une description élémentaire parmi celles que le lexique propose pour former ce que nous appellerons un *étiquetage syntaxique* de la phrase. La réunion des descriptions composant un étiquetage complétée par des relations de préséance exprimant l'ordre des mots dans la phrase va former une description d'arbre unique point de départ du processus d'analyse syntaxique proprement dit. Ce dernier va consister à itérer l'opération de neutralisation de traits opposés pour spécifier progressivement la description. L'analyse réussira si elle s'achève par un arbre où tous les traits ont été neutralisés.

3 - L'analyse syntaxique électrostatique

Si la notion de description d'arbres permet une souplesse très grande et une puissance d'expression, la contrepartie toutefois est que, d'un point de vue computationnel, l'analyse syntaxique basée sur les descriptions d'arbres peut être très coûteuse [KNT01]. Naïvement, analyser une phrase consiste à trouver tous les modèles d'une description associée par un lexique à la phrase. La recherche de tels modèles est hautement indéterministe. Dans les formalismes réalistes basés sur les descriptions d'arbres, cet indéterminisme est limité par une discipline très contraignante dans le mécanisme de composition syntaxique.

Dans les grammaires d'interaction, c'est le mécanisme de neutralisation de polarités qui assure cette discipline. La notion de polarités pour contrôler l'analyse syntaxique a déjà été utilisée par D. Duchier et S. Thater [DT99] avec la différence que, dans leur approche, les polarités sont attachées aux syntagmes. En attachant les polarités aux traits morpho-syntaxiques, les grammaires d'interaction autorisent seulement plus de subtilité dans le pouvoir d'expression du formalisme. Maintenant, comment utilisons nous les polarités dans l'analyse ? Nous exploitons les polarités en deux temps pour réduire les deux sources d'indéterminisme qui résident dans le choix d'un étiquetage syntaxique de la phrase à analyser et dans le choix des paires de traits duaux à neutraliser dans le processus d'analyse proprement dit.

- Dans la phase d'étiquetage, nous éliminerons les étiquetages incorrects en utilisant un principe de *neutralité globale* des traits polarisés présents dans la description d'arbre associée à un étiquetage donné de la phrase : durant le processus d'analyse syntaxique, pour chaque valeur de trait, la somme des polarités est constante du fait que l'opération de neutralisation de traits laisse celle-ci inchangée et si l'analyse réussit, cette somme doit être égale à zéro. Nous appelons l'application de ce principe *étiquetage électrostatique*.
- La phase d'analyse syntaxique proprement dite est accomplie de façon incrémentale en parcourant la phrase dans l'ordre de lecture. Le nombre de neutralisations potentielles est réduit de façon significative en appliquant des heuristiques basées sur des considérations psycholinguistiques (analyse syntaxique incrémentale, mémoire de travail limitée ...). Ces heuristiques ont été inspirées par les travaux de [Joh98, Mor00] menés dans le cadre des grammaires catégorielles. En particulier, nous limitons le nombre de *nœuds syntaxiques actifs*, c'est-à-dire porteurs de traits non neutres, afin de contraindre les neutralisations de traits à se réaliser le plus tôt possible. Nous appelons l'application de telles heuristiques *analyse syntaxique électrostatique contrôlée*.

4 - L'intégration de la sémantique : les grammaires d'interaction synchrones

La syntaxe des langues n'est pas un but en soi mais un moyen pour accéder à la sémantique et on ne peut pas concevoir un formalisme syntaxique sans se demander comment il va nous permettre d'accéder à la sémantique. Les grammaires catégorielles ont une réponse particulièrement élégante à cette question qui fait appel à la sémantique de Montague [Car98]. La dénotation d'une phrase est exprimée sous forme d'un λ-terme typé d'une logique d'ordre supérieure qui représente une fonction de la dénotation des mots de la phrase. Ce qui est important est que cette fonction est une projection de la structure syntaxique de la phrase qui s'obtient en oubliant dans cette structure l'ordre linéaire entre les syntagmes. La conséquence de cette dépendance étroite de la sémantique vis-à-vis de la syntaxe et de l'absence d'autonomie de la première vis-à-vis de la seconde est un alourdissement artificiel de la syntaxe pour rendre compte d'aspects particuliers de la sémantique comme par exemple de la portée des quantificateurs. Tout doit être dit dans la syntaxe et la sémantique n'en

est qu'un reflet passif.

Les grammaires d'interaction visent à établir un lien beaucoup plus souple entre syntaxe et sémantique si bien que cette dernière puisse être réellement autonome par rapport à la première et réagir sur elle. En ajoutant une dimension sémantique aux grammaires d'interaction, on obtient un formalisme qui englobe à la fois syntaxe et sémantique et que nous baptisons *grammaires d'interaction synchrones*. Les grammaires d'interaction synchrones comporte deux niveaux : un niveau syntaxique qui est celui qui a été présenté jusqu'à maintenant et un niveau sémantique qui utilise aussi la notion de traits polarisés et celle de descriptions mais au lieu de descriptions d'arbres, il s'agit de descriptions de graphes acycliques orientés (DAG pour Directed Acyclic Graphs). Ces DAG représentent les dénotations des énoncés sous de forme de relations de dépendances entre prédicats et arguments, ces arguments pouvant être des individus ou eux-mêmes des prédicats. Les descriptions de DAG représentent des DAG sous-spécifiés et c'est le même mécanisme de neutralisation de traits (sémantiques cette fois) qu'au niveau syntaxique qui va permettre de spécifier progressivement ces descriptions pour en construire les modèles. La synchronisation entre syntaxe et sémantique est effectuée par une fonction qui associe chaque nœud syntaxique à au plus un nœud sémantique. Cette contrainte est particulièrement souple et laisse une très grande liberté relative des deux niveaux, l'un par rapport à l'autre.

5 - L'organisation modulaire et hiérarchisée des grammaires d'interaction

La seule façon d'évaluer la pertinence d'un formalisme linguistique est de le confronter à des corpus de taille réelle mais pour un formalisme comme les grammaires d'interaction où les ressources grammaticales ne sont pas apprises automatiquement, cela pose le problème de la construction de grammaires à large couverture. Ces grammaires ont nécessairement des tailles importantes et la factorisation de l'information y est cruciale tant pour des raisons de lisibilité linguistique que de maintenance de la cohérence globale. Les travaux de M.-H. Candito [Can99] ont permis d'avancer dans l'organisation des grammaires basées sur les descriptions d'arbres avec la notion de *méta-grammaire*. Dans une méta-grammaire, deux types de mécanisme permettent de factoriser l'information : l'héritage et le croisement. La méta-grammaire est organisée en modules selon une hiérarchie grâce à une relation d'héritage. Les modules qui vont servir à construire la grammaire proprement dite sont les modules terminaux de cette hiérarchie mais, pour cela, ils vont devoir être croisés entre eux.

A partir de ces travaux, nous essaierons d'aller plus loin dans la même direction. Nous proposerons une présentation beaucoup plus formalisée et déclarative des méta-grammaires et surtout nous proposerons un mécanisme d'ancrage lexical des grammaires générées beaucoup plus souple qui fait appel à la notion de profil morpho-syntaxique : à chaque module de la méta-grammaire sera associée une structure de traits qu'on appellera son profil et qui décrira les propriétés morpho-syntaxiques des mots pouvant ancrer le module. Bien entendu, nos idées seront appliquées aux grammaires d'interaction mais elles

peuvent être étendues à tout formalisme utilisant des descriptions d'arbres.

6 - Comparaison avec d'autres formalismes et perspectives

En conclusion, nous comparerons les grammaires d'interaction avec des formalismes qui sont en rapport avec elles sous divers angles : les grammaires de propriétés de P. Blache [Bla01] qui sont fondées sur la vision d'une grammaire comme système de contraintes et les grammaires minimalistes de E. Stabler [Sta97] qui utilisent la neutralisation de traits polarisés pour contrôler le processus de dérivation syntaxique.

Nous situerons aussi les grammaires d'interaction par rapport à des formalismes ou des familles de formalismes déjà bien installés, les TAG [AR01], les grammaires de dépendance [Mel88, Hud90] et HPSG [PS94]. A partir de cela, nous indiquerons quelques perspectives d'avenir.

Chapitre 1

Des réseaux de démonstration aux grammaires d'interaction

Dans ce chapitre, nous décrirons le chemin qui, partant de la présentation logique des grammaires catégorielles à l'aide du calcul de Lambek, nous a amené à concevoir le modèle des grammaires d'interaction grâce aux travaux menés sur la logique linéaire et plus particulièrement sur les réseaux de démonstration.

1.1 La rigidité des grammaires de Lambek

Le calcul de Lambek[Lam58, Lam61] a fait faire un pas important aux grammaires catégorielles dans leur formalisation théorique. Des règles ad hoc avaient dû être ajoutées aux règles de base des grammaires AB[Moo88] pour couvrir un certain nombre de phénomènes syntaxiques. Le calcul de Lambek a permis d'unifier toutes ces règles dans un système logique où elles sont devenues des théorèmes. Le corollaire a été que l'analyse syntaxique des langues a pu être vue autrement qu'un calcul algébrique et s'est trouvée ramenée à un cas particulier de déduction logique.

Une grammaire de Lambek est définie par la donnée d'un certain nombre de types syntaxiques de base qui sont des formules logiques atomiques. Puis, à l'aide des implications gauche ($\backslash$) et droite ($/$), on construit des types complexes qui sont des formules logiques composées du calcul de Lambek. La signification linguistique de ces deux implications est alors la suivante : si ω_1 est une expression de la langue de type A/B et si ω_2 est une autre expression de type B, alors la concaténation $\omega_1.\omega_2$ des deux expressions est une expression de type B ; nous avons l'interprétation symétrique pour l'implication gauche. Un lexique associe à chaque mot de la langue un ensemble fini de types syntaxiques et l'analyse syntaxique d'une phrase revient à une déduction logique. Soit à analyser une phrase $\omega_1 \ldots . \omega_n$. Après avoir sélectionné dans le lexique une catégorie syntaxique A_i pour chaque mot ω_i de la phrase, l'analyse consiste à déduire des hypothèses $A_1, \ldots, A_n$, prises dans cet ordre, la formule S, où S représente le type phrase[1].

[1]Dans la suite du texte, les symboles représentant les types syntaxiques seront des abréviations de mots anglais. ainsi S correspond à *sentence*.

Considérons par exemple la grammaire formée des types de base suivants : N(nom), NP(syntagme nominal) et S(phrase). Les types complexes formés à partir de ces trois types sont :
$Det = NP/N$ (*déterminant*)
$IntrV = NP\backslash S$ (*verbe intransitif*)
$TrV = IntrV/NP = (NP\backslash S)/NP$ (*verbe transitif*)
$NrModif = N\backslash N$ (*modifieur droit de nom*)
$ObjRelPro = NrModif/(S/NP) = (N\backslash N)/(S/NP)$ (*pronom relatif objet*)
Nous voulons analyser la phrase *« la personne que Jean voit travaille »* à l'aide d'une grammaire de Lambek. Cette grammaire est définie par un lexique qui utilise les types qui viennent d'être définis et qui associe respectivement aux mots de la phrase les types Det, N, $ObjRelPro$, NP, TrV, $IntrV$. Cela revient à déduire dans le calcul de Lambek la formule S des hypothèses Det, N, $ObjRelPro$, NP, TrV, $IntrV$, c'est-à-dire en remplaçant les formules composées par leur définition :

$$NP/N,\ N,\ (N\backslash N)/(S/NP),\ NP,\ (NP\backslash S)/NP,\ NP\backslash S$$

Si nous nous plaçons dans le cadre de la déduction naturelle, la démonstration qui réalise cette déduction se présente ainsi :

$$\dfrac{\dfrac{NP/N \qquad \dfrac{\dfrac{N \qquad \dfrac{(N\backslash N)/(S/NP) \qquad \dfrac{\dfrac{\dfrac{(NP\backslash S)/NP \quad [NP]}{NP \qquad NP\backslash S}(\backslash_E)}{S}(/_I)}{S/NP}(/_E)}{N\backslash N}(\backslash_E)}{N}(/_E)}{NP} \qquad NP\backslash S}{S}(\backslash_E)$$

Plus généralement, pour définir les types syntaxiques, on a besoin essentiellement des implications gauche et droite et on peut se passer de la conjonction. Avec cette restriction, les démonstrations nécessitent seulement quatre règles d'inférence : les deux règles d'introduction et les deux règles d'élimination des implications gauche et droite. Le progrès par rapport aux grammaires catégorielles telles qu'on les connaissait auparavant est évident.

Cela ne veut pas dire pour autant que le pouvoir d'expression du formalisme s'en trouve accru. Un facteur important de rigidité du calcul de Lambek réside dans le fait qu'il combine de façon indissociable deux propriétés : la linéarité qui fait que, dans les démonstrations, les formules logiques se comportent comme des ressources consommables et la non commutativité qui fait que, dans les démonstrations toujours, il y a un ordre cyclique entre les formules qui y apparaissent. On ne peut pas séparer les deux propriétés. Lorsque l'on écrit A/B, on écrit que l'on attend une et une seule occurrence de B pour obtenir A (linéarité) et que l'on attend cette occurrence à droite (non commutativité). Ceci limite fortement le pouvoir d'expression du formalisme. Par exemple, le type $ObjRelPro$ tel qu'il vient d'être défini par $(N\backslash N)/(S/NP)$

exprime qu'un pronom relatif attend une proposition à un mode conjugué à droite (S) à qui il manque un syntagme nominal (NP) à sa périphérie droite. Ce syntagme nominal correspond à l'objet extrait de la relative, si l'on considère que le syntagme nominal qui est l'antécédent de la relative était dans une position initiale à l'intérieur de la relative et qu'il a été déplacé. Or, en français et dans bien d'autres langues, cet objet peut très bien être extrait non pas de la périphérie de la proposition relative mais de l'intérieur (extraction médiane) comme dans la phrase *« la personne que Jean voit aujourd'hui travaille »* où la trace de *la personne* dans la relative *« que Jean voit aujourd'hui »* se situe entre *voit* et *aujourd'hui*. Le calcul de Lambek est incapable de rendre compte des extractions médianes.

Une première réponse à cette rigidité consiste à enrichir le calcul de Lambek d'opérateurs logiques particuliers qui permettent de relâcher d'une manière contrôlée la stricte non commutativité initiale. C'est cette voie qu'ont suivi en particulier G. Morrill [Mor94] et M. Moortgat [Moo96]

Nous avons choisi une autre voie qui doit beaucoup aux travaux autour de la théorie de la démonstration en logique linéaire. L'introduction de la logique linéaire [Gir87] a permis de faire le lien entre le calcul de Lambek et la logique dans ses formes classique et intuitionniste. La logique linéaire apparaît comme un raffinement de ces deux formes de logique et le calcul de Lambek apparaît comme une version non commutative et restreinte de la logique linéaire intuitionniste. La restriction consiste à ne considérer que le noyau multiplicatif de la logique linéaire.

Pour dissocier la linéarité de la non commutativité, nous avons choisi de conserver une modélisation logique de la première à l'intérieur de la logique linéaire commutative et de renvoyer le traitement de l'ordre des mots qui correspond à la non commutativité du calcul de Lambek à un niveau algébrique, celui d'un monoïde non commutatif de termes représentant les expressions de la langue [Per99]. La logique linéaire intervient alors comme un système de types sur ce monoïde.

Ce qui nous intéresse pour représenter les types syntaxiques, ce n'est pas toute la logique linéaire, mais seulement le noyau multiplicatif de la logique linéaire intuitionniste et même réduit au fragment contenant comme seul opérateur l'implication linéaire ($\multimap$). La séparation entre linéarité et ordre des mots ne change pas le paradigme de l'analyse syntaxique vue comme déduction, ce qui nous amène à étudier la théorie de la démonstration dans ce cadre logique particulier.

1.2 Les réseaux de démonstration de la logique linéaire intuitionniste

Le fait qu'en logique linéaire multiplicative, toute formule qui apparaît dans une démonstration y est utilisée exactement une fois fait qu'on peut y tracer son histoire et permet de représenter la démonstration sous forme d'un réseau. Les entrées-sorties du réseau représentent les hypothèses et conclusions de la démonstration. En logique linéaire classique, ces réseaux présentent une parfaite symétrie si bien que toute conclusion de type A peut être vue comme

une hypothèse du type dual $A^\perp$, où $A^\perp$ représente la négation linéaire de A, et inversement. Le passage de la logique linéaire classique à la logique linéaire intuitionniste ne fait que figer l'orientation des réseaux dans une configuration comportant une seule conclusion, toutes les autres entrées-sorties étant des hypothèses.

Nous allons maintenant nous intéresser plus précisément au fragment implicatif de la logique linéaire intuitionniste multiplicative que nous nommerons IILL. Ce fragment nous intéresse ici car c'est lui qui permet de représenter l'essentiel des types syntaxiques utilisés pour les grammaires que nous aurons à définir. Dans IILL, les formules sont construites à partir de la seule implication linéaire ($\multimap$) comme connecteur. Si on utilise le calcul des séquents pour exprimer ce fragment logique, les séquents ont la forme $F_1, \ldots, F_n \vdash G$ où $F_1, \ldots, F_n, G$ sont des formules atomiques ou composées uniquement avec $\multimap$ et où n peut être nul (séquents avec partie gauche vide). Le caractère intuitionniste de la logique est garanti par la présence d'une et une seule formule dans la partie droite des séquents.

Le système déductif correspondant comporte deux règles d'inférence représentant l'introduction de l'implication linéaire à gauche ou à droite des séquents. Ce système peut être traduit dans le système de séquents unilatères sans partie gauche de la logique linéaire multiplicative classique CMLL[2] en utilisant des polarités. Deux fonctions $()^+$ et $()^-$ transforment les formules d'IILL en formules polarisées de CMLL. Ici, nous n'utiliserons pas cette traduction et nous conserverons les formules et les séquents dans leur forme initiale.

En logique linéaire, une démonstration se présente sous forme d'un réseau donc démontrer un séquent consiste à construire un réseau de démonstration de ce séquent. Dans IILL, la démonstration d'un séquent $F_1, \ldots, F_n \vdash G$ consiste d'abord à déplier l'arbre syntaxique de chacune des hypothèses $F_1, \ldots, F_n$, qui vont constituer les entrées du réseaux et de la conclusion G, qui va en constituer la sortie. Dans ce dépliage, chaque sous-formule va être polarisée soit en formule d'entrée, soit en formule de sortie. Dans une démonstration présentée comme un calcul de séquents, chaque sous-formule d'entrée serait produite dans la partie gauche d'un séquent alors que chaque sous-formule de sortie serait produite dans la partie droite du séquent. Dit d'une autre façon, les sous-formules de sortie représentent les buts intermédiaires à atteindre et les sous-formules d'entrée les ressources utilisées pour cela. Dans chaque formule dépliée, nous allons avoir deux types de liens d'inférence, qui sont représentés à droite sur la figure 1.1[3] Si on observe les deux types de liens d'inférence, on constate dans une formule dépliée une alternance de sous-formules d'entrée et de sortie car l'argument d'une sous-formule d'entrée est une formule de sortie (lien $\multimap$ entrée) et, inversement, l'argument d'une sous-formule de sortie est une formule d'entrée (lien $\multimap$ sortie).

Il est facile d'établir une correspondance entre les liens d'inférence des réseaux de démonstration et ceux des arbres de démonstration en déduction naturelle :

[2]Dans CMLL, la syntaxe des formules logiques est la suivante : $F ::= A|A^\perp|F \otimes F|F \parr F$, où A représente une formule atomique quelconque.

[3]Les sous-formules de sortie sont représentées par des arcs orientés vers le bas et les sous-formules d'entrées le contraire.

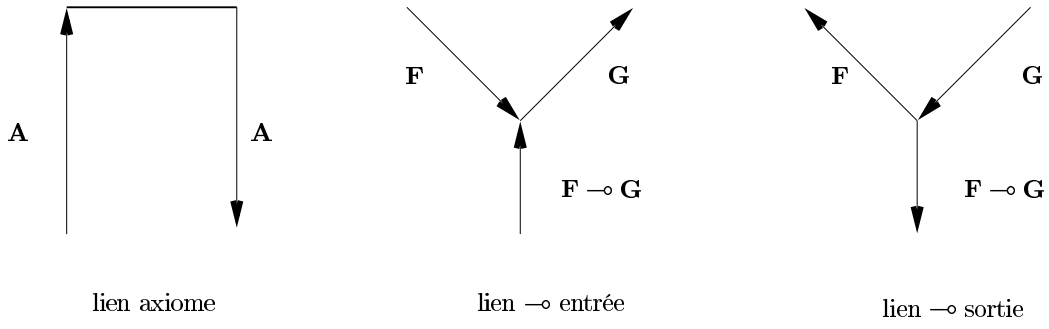


FIG. 1.1 – Types de liens des réseaux de démonstration d'IILL

- Un lien $\multimap$ entrée correspond à une application de la règle d'élimination de l'implication linéaire. En entrée du lien, il y a l'implication $F \multimap G$ et F et en sortie, il y a G.
- Un lien $\multimap$ sortie correspond à une application de la règle d'introduction de l'implication linéaire. En entrée, on trouve G et en sortie $F \multimap G$. L'autre formule en sortie F représente l'hypothèse qui est déchargée dans l'inférence. Contrairement à la logique intuitionniste, il y a une correspondance biunivoque entre inférences d'introduction de l'implication linéaire et hypothèses déchargées, ce que traduit ce lien.

A l'aide de l'isomorphisme de Curry-Howard, on peut encore interpréter les liens $\multimap$ entrée et sortie respectivement comme des applications et des abstractions d'un λ-calcul linéaire associé aux démonstrations.

La structure obtenue en dépliant les arbres syntaxiques des formules d'un séquent d'IILL sera appelée *structure de démonstration initiale* d'IILL. Une démonstration consiste ensuite à relier les formules atomiques duales par paires à l'aide de liens axiomes, représentés à gauche sur la figure 1.1. Nous utiliserons l'expression *structure de démonstration* sans préciser plus pour parler d'une structure de démonstration initiale dans laquelle certains liens axiomes ont éventuellement été posés. Lorsque toutes les formules atomiques d'une structure de démonstration sont connectées par des liens axiomes, nous parlerons de *structure de démonstration finale*. A partir d'une structure de démonstration initiale, il n'est pas toujours possible d'obtenir une structure de démonstration finale mais lorsque c'est le cas, cela ne veut pas dire pour autant que l'on a obtenu une démonstration correcte. Pour mériter le nom de *réseau de démonstration*, elle doit vérifier certains critères.

Il y a plusieurs critères possibles et nous allons en présenter deux qui ont un double intérêt : ils utilisent le caractère spécifique des réseaux intuitionnistes, c'est-à-dire la polarisation des formules, et, par rapport à la modélisation linguistique qui motive notre étude, ils peuvent donner lieu facilement à interprétation linguistique. Le premier, qui est dû à F. Lamarche [Lam96], s'appuie sur les chemins de parcours induits dans les réseaux par la polarisation des formules ; le second, qui est dû à P. de Groote [dG99], est très proche du premier et utilise un étiquetage des réseaux par les termes d'un monoïde commutatif.

1.2.1 Correction des réseaux de démonstration fondée sur les chemins de parcours

La polarisation des formules dans les réseaux d'IILL déterminent des chemins de parcours qui vont des entrées vers la sortie. D'une certaine façon, ces chemins permettent de reconstruire à partir d'un réseau l'arbre de démonstration correspondant en déduction naturelle. Ils sont déterminés par la règle de parcours suivante :

> *on entre dans un lien par une prémisse qui est une formule de sortie ou une conclusion qui est une formule d'entrée et on en sort par une prémisse qui est une formule d'entrée ou une conclusion qui est une formule de sortie.*
>
> *La seule restriction est que l'on s'interdit de sortir d'un lien $\multimap$ sortie par sa prémisse qui est une formule d'entrée.*

Cette dernière restriction tient au fait que la prémisse en question joue le rôle d'hypothèse déchargée et qu'elle ne peut être que le début d'un chemin. Ceci étant dit, le critère de correction est très simple :

> *L'ensemble des chemins de parcours doit former un arbre, l'arbre de déduction naturelle, et tout chemin qui part de la prémisse d'un lien $\multimap$ sortie et qui est une formule d'entrée doit passer par l'autre prémisse.*

La seconde condition correspond au fait qu'en déduction naturelle, on ne peut décharger qu'une hypothèse ayant servi à démontrer la formule que l'on veut abstraire. Dit d'une autre façon, le critère signifie qu'un chemin maximum part d'une hypothèse, déchargée ou pas, et termine à la conclusion du réseau et, s'il part d'une hypothèse déchargée, il doit passer forcément par la formule abstraite correspondante.

Le critère de correction, tel qu'il vient d'être défini, se trouve validé par la proposition suivante :

Proposition 1.2.1. *Un séquent $F_1, \ldots, F_n \vdash G$ de IILL est démontrable ssi il existe un réseau de démonstration de ce séquent au sens de F. Lamarche.*

Prenons un exemple. Soit à démontrer le séquent d'IILL :

$$a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \ \vdash\ a \multimap e$$

En dépliant les arbres syntaxiques des formules du séquent, on obtient la structure de démonstration initiale de la figure 1.2. Il s'agit maintenant de relier les formules atomiques par paires duales à l'aide de liens axiomes. Il n'y a ici qu'une façon de le faire, celle qui est proposée par la figure 1.3. On vérifie alors facilement que le réseau obtenu est correct. En déconnectant les hypothèses déchargées des liens $\multimap$ sortie auxquels elles se rattachent, on obtient exactement l'arbre de déduction naturelle sous-jacent, tel que le montre la figure 1.4[4]. C'est la correction du réseau qui rend possible l'existence de cet arbre.

[4]Les hypothèses déchargées sont notées entre crochets.

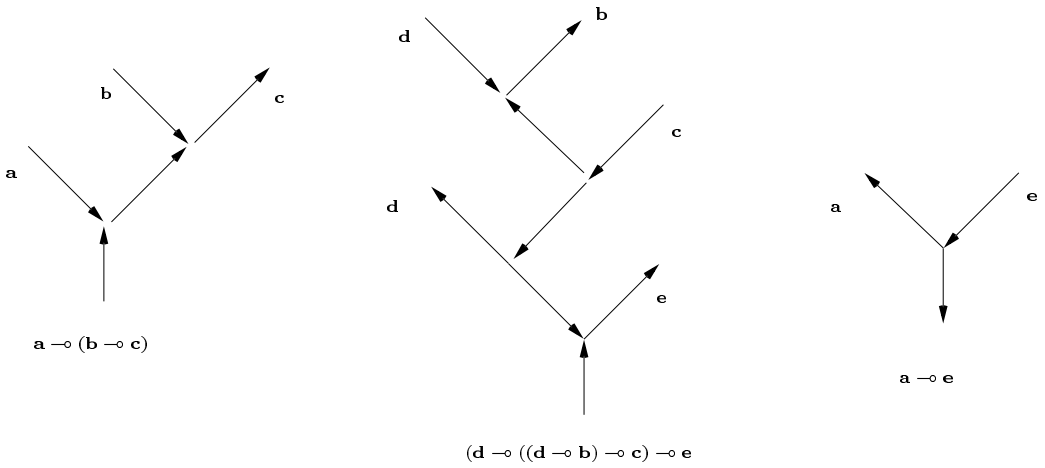


FIG. 1.2 – Structure de démonstration initiale du séquent $a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \vdash a \multimap e$

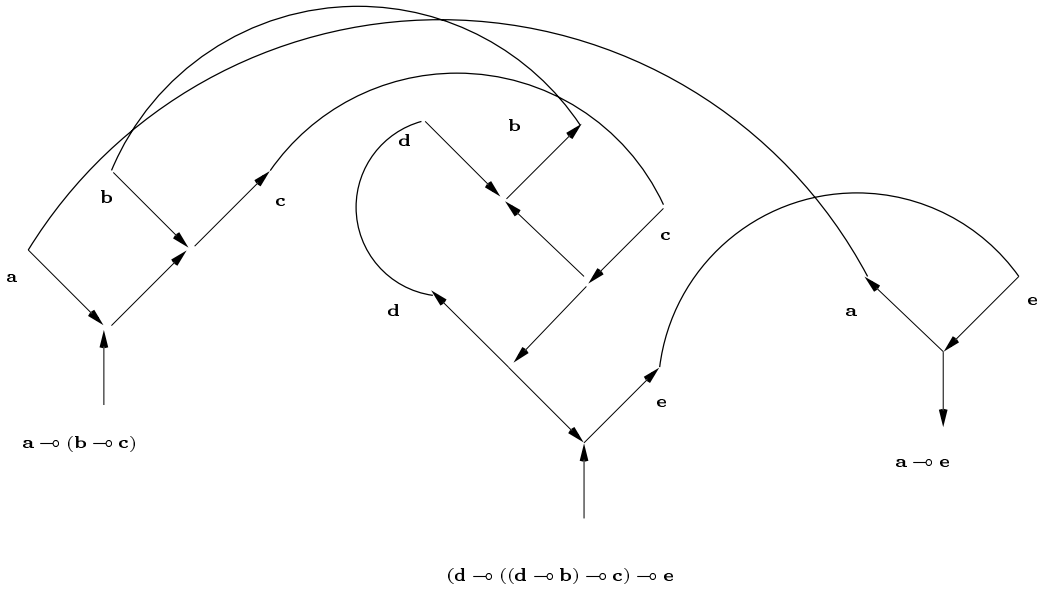


FIG. 1.3 – Réseau de démonstration du séquent $a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \vdash a \multimap e$

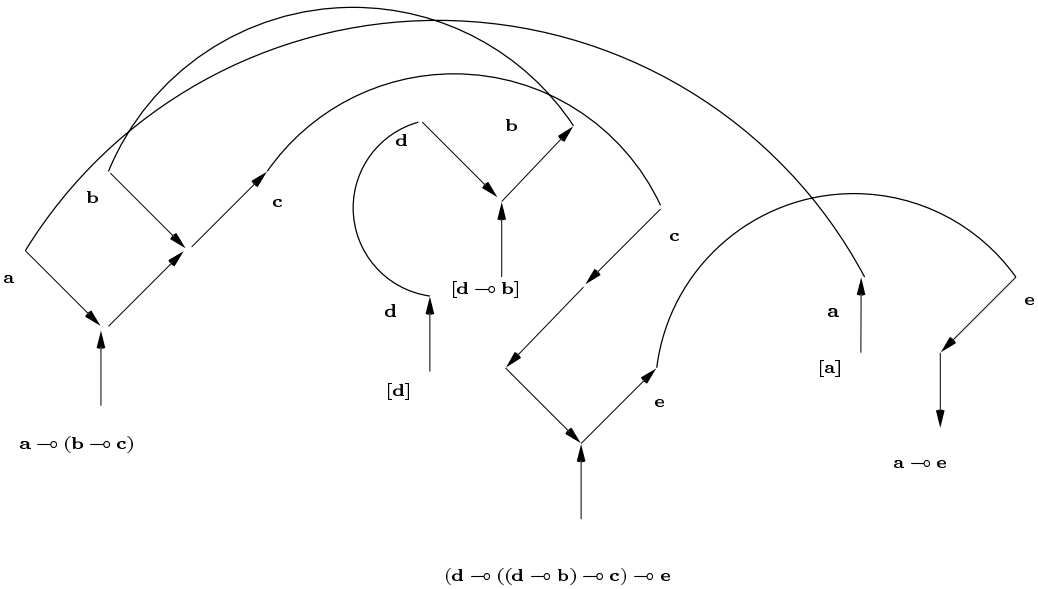


FIG. 1.4 – Arbre de déduction naturelle du séquent $a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \vdash a \multimap e$

1.2.2 Correction des réseaux de démonstration fondée sur l'étiquetage par un monoïde commutatif

L'étiquetage des réseaux de démonstration intuitionnistes par un monoïde commutatif fut introduit par D. Roorda [Roo91] mais ce dernier n'était pas certain que cet étiquetage soit suffisant pour garantir la correction des réseaux. Cet étiquetage s'appuie directement sur la sémantique des phases de la logique linéaire [Gir87] et c'est P. de Groote qui a montré qu'il pouvait servir de critère de correction [dG99].

On considère un monoïde commutatif avec un ensemble dénombrable d'éléments premiers. Ils sont notés α, β, $\gamma \ldots$ et l'opération du monoïde est notée $+$. On étiquette chaque sous-formule d'un réseau par un élément du monoïde en respectant une loi qu'on appellera la *loi des nœuds* par analogie avec la loi des nœuds pour un courant électrique circulant dans un réseau :

> *Pour chaque lien du réseau, la somme des étiquettes des formules mv aboutissant au lien est égale à la somme des étiquettes des formules qui en sont issues.*

Il n'est pas toujours possible d'étiqueter une structure de démonstration finale d'IILL en respectant cette loi. Mais en plus, l'existence d'un tel étiquetage ne garantit la correction du réseau que si les étiquettes des hypothèses sont indépendantes, ce qu'établit la proposition suivante :

Proposition 1.2.2. *Un séquent $F_1, \ldots, F_n \vdash G$ d'IILL est démontrable ssi il existe un réseau étiqueté de ce séquent tel que les étiquettes des hypothèses déchargées ou pas sont premières entre elles deux à deux.*

Ce critère de correction peut être utilisé pour concevoir des algorithmes originaux de démonstration dans IILL. Reprenons la démonstration du séquent

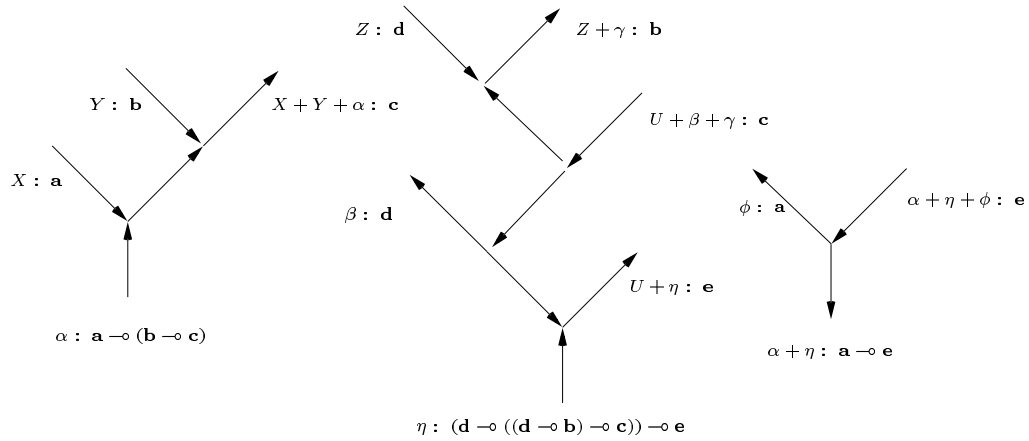


FIG. 1.5 – Structure de démonstration initiale étiquetée du séquent $a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \vdash a \multimap e$

$a \multimap (b \multimap c),\ (d \multimap ((d \multimap b) \multimap c)) \multimap e \vdash a \multimap e$. En dépliant ses formules, on obtient la structure de démonstration initiale de la figure 1.2. Nous étiquetons ensuite les sous-formules de cette structure selon la procédure suivante :

- on étiquette les hypothèses par des éléments premiers tous différents ;
- on étiquette les prémisses des liens $\multimap$ entrée qui sont des formules de sortie par des variables toutes différentes (notées par des majuscules) ;
- on étiquette la conclusion générale par la somme des étiquettes des hypothèses non déchargées ;
- on propage les étiquettes à toutes les sous-formules en appliquant la loi des nœuds.

En appliquant cette procédure à notre exemple, nous obtenons la structure de démonstration initiale étiquetée de la figure 1.5. Nous avons seulement noté les étiquettes des racines et des feuilles des arbres syntaxiques des formules. Il reste ensuite à relier les feuilles par paires duales à l'aide de liens axiomes. Ce sont alors les étiquettes des feuilles qui vont garantir que le réseau obtenu est correct. Lorsqu'on relie deux feuilles duales par un lien axiome, leurs étiquettes sont unifiées. En conséquence, la démonstration dans IILL, se ramène à un problème d'unification de termes d'un monoïde commutatif. On montre qu'on peut même toujours choisir une stratégie où un des termes à unifier est clos, si bien que le problème se réduit à un problème de filtrage.

Dans notre exemple, le typage des feuilles de la structure de démonstration initiale n'offre qu'une seule possibilité de poser des liens axiomes et on vérifie que l'unification des termes associés est possible avec la substitution suivante :$\{X \leftarrow \phi,\ Y \leftarrow \beta+\gamma,\ Z \leftarrow \beta,\ U \leftarrow \alpha+\phi\}$.

Ce critère peut être adapté sans problème à la version non commutative d'IILL, c'est-à-dire le fragment implicatif du calcul de Lambek : il suffit de remplacer le monoïde commutatif par un monoïde non commutatif.

1.3 Des grammaires fondées sur la logique linéaire intuitionniste

Nous avons vu dans la section 1.1 comment les grammaires de Lambek interprètent la logique pour exprimer la syntaxe des langues : les formules logiques représentent les catégories grammaticales de la langue et les démonstrations la syntaxe des phrases. Le cadre logique est alors celui du calcul de Lambek et on peut y représenter les démonstrations plutôt sous forme de réseaux que d'arbres de déduction naturelle.

Cette interprétation se transpose sans problèmes des réseaux de Lambek aux réseaux d'IILL étiquetés, tels qu'ils viennent d'être présentés. Il faut bien entendu donner une interprétation aux étiquettes. Il est alors naturel de considérer le monoïde non commutatif des expressions phonologiques formées à l'aide des mots de la langue. L'opération du monoïde est alors la concaténation d'expressions qui est notée « . ». La correction d'un réseau au sens de P. de Groote est assurée en faisant abstraction de l'ordre dans le monoïde. Ceci permet d'exprimer les propriétés de linéarité de la langue. Pour exprimer les propriétés relatives à l'ordre des mots, on utilisera l'ordre dans les termes phonologiques étiquetant les réseaux. Voyons comment à travers un exemple.

Reprenons la phrase *« la personne que Jean voit aujourd'hui travaille »* qui illustre le phénomène d'extraction médiane qui n'est pas représentable à l'aide du calcul de Lambek. Le lexique définissant une grammaire de Lambek particulière associe à chaque mot de la langue un ensemble de types syntaxiques sous forme de formules de Lambek. Nous allons assouplir le formalisme en enrichissant le pouvoir d'expression du lexique de la façon suivante : au lieu d'associer à un mot des types, nous associons les arbres syntaxiques de ces types étiquetés par des expressions de langue en respectant les contraintes suivantes :

- les prémisses des liens $\multimap$ sortie qui sont des formules d'entrée sont étiquetées par des mots spéciaux tous différents qui sont ajoutés aux mots de la langue ; ils sont notés par les minuscules *x, y, z*, éventuellement indicées et représentent en général les traces de syntagmes déplacés ;
- les prémisses des liens $\multimap$ entrée qui sont des formules de sortie sont étiquetées par des variables seules ou concaténées ; elles sont notées par des majuscules *X, Y, Z*, éventuellement indicées ;
- lorsque l'on fait abstraction de l'ordre des mots dans les expressions phonologiques, tous les liens doivent respecter la loi des nœuds.

La figure 1.6 présente des entrées lexicales possibles pour les mots de la phrase *« la personne que Jean voit aujourd'hui travaille »*. L'adverbe *aujourd'hui* est un modifieur de phrase et nous nous restreindrons à son utilisation en fin de phrase. L'entrée lexicale du pronom relatif *que* montre comment est modélisée l'extraction médiane. Le terme *x1* représente la trace de l'objet extrait, *la personne*, et le terme *X2.x1.X3* représente la proposition relative privée de son pronom relatif. La trace y occupe bien une position potentiellement médiane selon les valeurs que prendront *X2* et *X3*. Sur la figure 1.6, en plus des entrées lexicales des mots de la phrase, on trouve la conclusion à laquelle la démonstration doit aboutir sous forme d'une formule de sortie S comme type de la

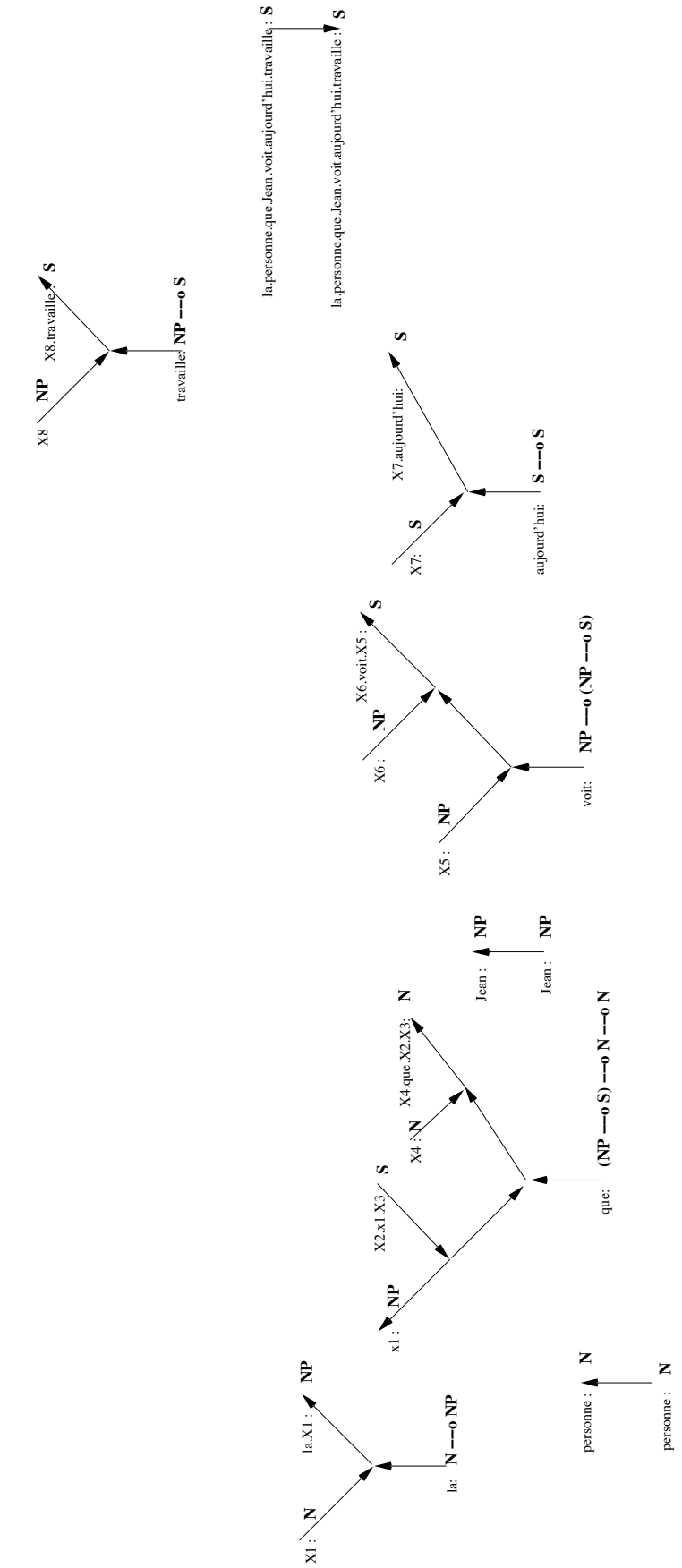


FIG. 1.6 – Entrées lexicales pour les mots de la phrase *« la personne que Jean voit aujourd'hui travaille »*

phrase *« la personne que Jean voit aujourd'hui travaille »*.

La forêt d'arbres syntaxiques représentée à la figure 1.6 constitue une structure de démonstration initiale. L'analyse syntaxique consiste ensuite à poser les liens axiomes sous la double contrainte des types syntaxiques et des termes phonologiques : les types syntaxiques doivent être les mêmes et les termes phonologiques doivent s'unifier. Dans notre exemple, une seule solution est possible qui correspond à la substitution suivante pour les variables phonologiques :

$$\begin{array}{lcl} \{\ X8 & \leftarrow & la.personne.que.Jean.voit.aujourd'hui, \\ X1 & \leftarrow & personne.que.Jean.voit.aujourd'hui, \\ X4 & \leftarrow & personne, \\ X5 & \leftarrow & x1, \\ X6 & \leftarrow & Jean, \\ X7 & \leftarrow & Jean.voit.x1, \\ X2 & \leftarrow & Jean.voit, \\ X3 & \leftarrow & aujourd'hui\ \} \end{array}$$

On obtient alors le réseau de la figure 1.7

On peut retrouver les grammaires de Lambek comme restrictions de ces grammaires plus souples en ajoutant des contraintes aux règles d'étiquetage des liens $\multimap$: pour les liens d'entrée, on n'autorise que la concaténation à gauche ou à droite d'étiquettes et, pour les liens de sortie que la soustraction à gauche ou à droite d'étiquettes.

1.4 La démonstration de théorèmes vue comme recherche de modèles de descriptions d'arbres

Cette idée découle du critère de correction des réseaux intuitionnistes de F. Lamarche [Lam96]. Les chemins de parcours dans un réseau tels qu'ils ont été définis dans la sous-section 1.2.1 induisent un ordre d'utilisation des formules atomiques du réseau. Cet ordre va permettre d'abstraire la notion de réseau de démonstration pour n'en conserver que le cœur, c'est-à-dire l'arbre formé par les relations d'ordre entre formules atomiques. Cet arbre peut être déduit de l'arbre de déduction naturelle sous-tendant le réseau mais il ne se confond pas avec lui. Par exemple, pour le réseau de la figure 1.3, on peut déduire de l'arbre de déduction naturelle correspondant, tel qu'il est représenté à la figure 1.4, l'arbre représentant l'ordre d'utilisation des formules atomiques de la figure 1.8. Sur cette figure, le fait qu'une formule atomique soit située plus haut qu'une autre dans l'arbre signifie qu'il y a un chemin de parcours de la première vers la seconde. La correction du réseau se traduit par le fait que la relation d'ordre entre formules atomiques définit un arbre avec certaines contraintes supplémentaires correspondant à la condition sur les hypothèses déchargées. Nous allons revoir la construction de réseaux dans IILL sous cet angle.

Nous avons vu qu'en dépliant l'arbre syntaxique des formules d'un séquent d'IILL, nous obtenions une structure de démonstration initiale et que la démonstration proprement dite se ramenait à connecter les formules atomiques par des liens axiomes en respectant un critère de correction.

Dans la structure de démonstration initiale, la relation d'ordre entre for-

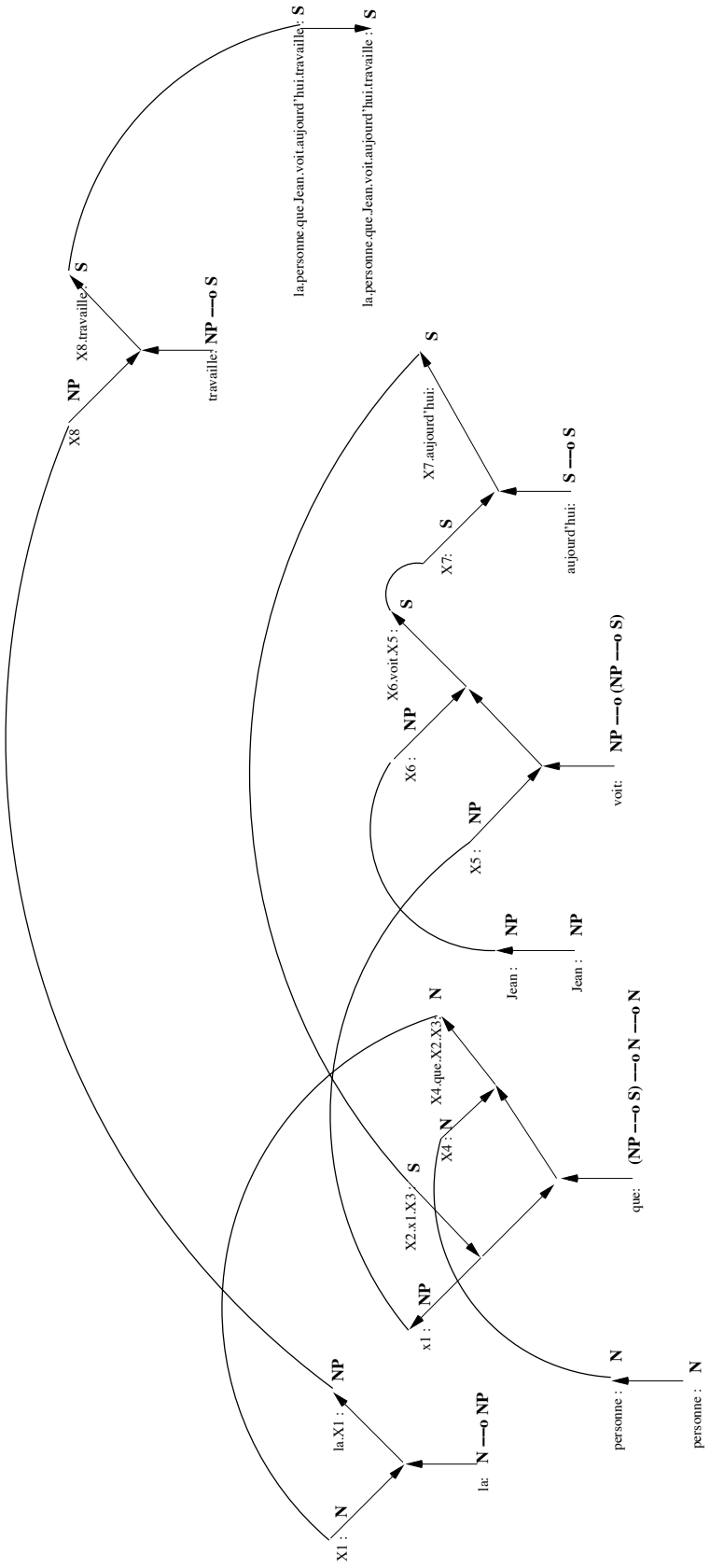


FIG. 1.7 – Analyse syntaxique de la phrase *« la personne que Jean voit aujourd'hui travaille »*

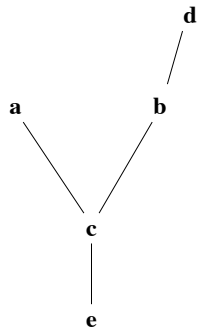


FIG. 1.8 – Arbre représentant l'ordre d'utilisation des formules atomiques dans le réseau de la figure 1.3

mules atomiques est partiellement spécifiée par les chemins de parcours déjà présents et c'est la pose des liens axiomes qui va permettre de la spécifier complètement. Le critère de correction de F. Lamarche introduit des contraintes quant à la façon de compléter cette spécification.

Ainsi dans la structure de démonstration initiale de la figure 1.2, les formules de sortie a et b du premier arbre syntaxique précèdent la formule d'entrée c. Dans le second arbre syntaxique, la formule d'entrée d ne précède aucune autre formule mais si nous voulons construire un réseau correct, nous devons construire un chemin de parcours qui part de la formule d'entrée d et qui passe par l'autre prémisse du lien $\multimap$ auquel d se rattache. Ce chemin passe nécessairement par la formule de sortie c. En résumé, le critère de correction nous impose que la formule d'entrée d précède la formule de sortie c et la pose des liens axiomes devra tenir compte de cette contrainte.

Nous allons formaliser cette représentation abstraite des réseaux de démonstration en utilisant la notion de *description d'arbre*. Cette notion n'est pas habituelle en théorie de la démonstration ; elle a été introduite en linguistique informatique pour exprimer l'ambiguïté syntaxique qui constitue une propriété importante des langues naturelles [MHF83, VS92]. L'idée est de représenter un arbre sous-spécifié ou un ensemble d'arbres par un ensemble de propriétés élémentaires le caractérisant. Le plus souvent, on trouve parmi ces propriétés des relations de domination et de préséance entre nœuds. L'intérêt de manipuler des descriptions d'arbres à la place d'arbres complètement spécifiés est que la représentation est plus compacte donc plus facile à traiter et que les descriptions peuvent se composer souplement entre elles. Voyons comment appliquer cette notion aux réseaux de démonstration.

A chaque séquent d'IILL, nous allons associer une description d'arbre qui est une représentation abstraite de la structure de démonstration initiale correspondante et qui est définie par les trois règles suivantes :

1. Les nœuds de la description sont les occurrences de formules atomiques du séquent. Chacun de ces nœuds est étiqueté par la formule atomique correspondante, précédée d'un signe + s'il s'agit d'une formule d'entrée et d'un signe – s'il s'agit d'une formule de sortie. Les signes sont contraires à l'habitude pratiquée en logique linéaire et suivent plutôt l'interprétation

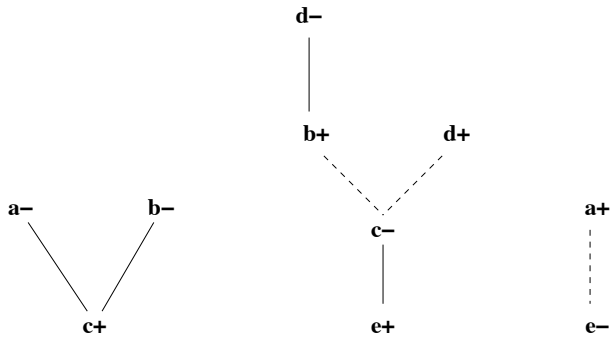


FIG. 1.9 – Description d'arbre traduisant la structure de démonstration initiale de la figure 1.3

linguistique qui en sera faite par la suite : une formule positive représentera une ressource disponible et une formule négative une ressource attendue.

2. Lorsqu'il existe un chemin de parcours d'une formule atomique vers une autre formule atomique dans l'arbre syntaxique d'une formule du séquent, nous le traduirons par une relation de parenté entre nœuds de la description correspondants.
3. Lorsqu'il n'existe pas de chemin de parcours d'une formule atomique à une autre mais que le critère de correction de F. Lamarche requiert la construction d'un tel chemin, nous le traduirons par une relation de domination large entre nœuds de la description correspondants.

Appliquons ces trois règles à la structure de démonstration initiale de la figure 1.2. Nous obtenons la description d'arbre de la figure 1.9.

Sous cet angle, comment va se traduire la construction du réseau proprement dite qui s'effectue par la pose de liens axiomes ? Connecter deux formules atomiques duales par un lien axiome dans un réseau en construction revient à identifier deux nœuds duaux, c'est-à-dire porteurs de la même formule mais avec des polarités opposées, dans la description correspondante. Pour le nœud résultant, il faut ensuite remplacer les deux étiquettes opposées par l'étiquette neutre correspondante. Nous appellerons cette opération *neutralisation élémentaire*.

La construction du réseau se réduit à spécifier de plus en plus la description de départ en itérant l'opération de neutralisation élémentaire entre nœuds duaux. Elle réussit si nous aboutissons à un arbre dans lequel toutes les étiquettes sont neutres et toutes les relations de domination large ont été réalisées sous forme d'un enchaînement de relations de parenté. Pour la description de la figure 1.9, en itérant 5 fois l'opération élémentaire de neutralisation entre nœuds, nous obtenons l'arbre de la figure 1.8.

Ce processus de construction pas à pas d'un arbre à partir d'une description par une suite de neutralisation élémentaires peut être présenté aussi d'une autre façon moins calculatoire sous l'angle de la théorie des modèles. Une description d'arbre traduisant un séquent d'IILL à démontrer peut être représenté sous forme d'une formule du premier ordre où les relations de parenté et de

domination large apparaissent sous forme de prédicats binaires. Cette formule décrit une spécification et le problème est de trouver un arbre qui soit un modèle de cette spécification. Bien entendu, ce modèle doit respecter des propriétés de neutralité et de minimalité mais nous ne nous attarderons pas sur cette façon de voir les choses ici : nous renvoyons le lecteur à l'article [Per01].

On pourrait effectuer une traduction analogue pour les réseaux de démonstration étiquetés tels qu'ils ont été définis à la sous-section 1.2.2. Nous obtiendrions des descriptions d'arbres étiquetées par les éléments d'un monoïde commutatif. L'opération de neutralisation élémentaire donnerait lieu alors à unification des étiquettes et cette dernière pourrait servir à guider les neutralisations.

Enfin, cette approche peut être étendue à IMLL tout entier. Le graphe des relations de préséance entre formules atomiques n'est alors plus un arbre mais un graphe acyclique orienté. Le lecteur pourra trouver les résultats relatifs à cette extension avec leur démonstration dans [Per01]. Il y trouvera aussi les démonstrations des résultats relatifs à IILL.

1.5 Les grammaires d'interaction primitives

Dans la section 1.3, nous avons défini un modèle linguistique qui s'appuie sur les réseaux de démonstration d'IILL. La traduction de la construction de réseaux en termes de recherche de modèles de descriptions d'arbres, telle qu'elle a été définie à la section précédente peut donc s'appliquer à ce modèle linguistique :

- Les syntagmes qui dans le modèle précédent étaient représentés par des occurrences de formules d'IILL exprimant leur type syntaxique seront maintenant représentés par les nœuds d'une description.
- Ceux qui étaient représentés par une formule d'entrée exprimaient des ressources disponibles ; les nœuds correspondants seront donc étiquetés par des catégories grammaticales positives.
- Ceux qui étaient représentés par une formule de sortie exprimaient des ressources attendues ; les nœuds correspondants seront donc étiquetés par des catégories grammaticales négatives.
- Une relation de parenté entre deux nœuds traduira le fait que l'un est un constituant immédiat de l'autre tandis qu'une relation de domination large traduira le fait que l'un est inclus au sens large dans l'autre sans que soit précisé le niveau de l'inclusion.

Dans le modèle linguistique précédent, l'ordre des mots était exprimé à l'aide de termes phonologiques étiquetant les réseaux. On pourrait traduire directement cet étiquetage par un étiquetage des nœuds des descriptions. Nous avons choisi une autre façon de faire, classique, puisqu'il s'agit d'utiliser des descriptions d'arbres ordonnés : pour cela, on introduit une relation binaire de préséance entre nœuds. Linguistiquement, cette relation exprimera le fait qu'un syntagme précède un autre syntagme dans la phrase où ils figurent tous les deux.

Si nous voulons conserver la correspondance entre description d'arbre et structure de démonstration, nous devons introduire des contraintes sur la forme syntaxique de nos descriptions. Formellement, ce que nous appellerons une

description syntaxique pour un ensemble de catégories grammaticales de base *Cat* sera défini de la façon suivante :

Définition 1.5.1. *Une description syntaxique est un ensemble de nœuds étiquetés par des éléments de* Cat*, éventuellement munis d'une polarité + ou − et un ensemble de relations de parenté, de domination large et de préséance entre ces nœuds. Les nœuds munis d'une polarité + seront dits positifs, les nœuds munis d'une polarité − seront dits négatifs et les autres seront dits neutres.*

De plus, elle vérifie les contraintes suivantes :

1. *tout nœud à la source d'une relation de parenté est positif ou neutre alors que la cible est négative ou neutre ;*
2. *tout nœud à la source d'une relation de domination large est négatif ou neutre alors que la cible est positive ou neutre ;*
3. *tout nœud est la cible d'au plus une relation de parenté et au plus une relation de domination large ;*
4. *tout nœud négatif est la source d'au moins une relation.*

Il est facile de montrer que toute description qui respecte ces conditions interprète au moins une structure de démonstration d'IILL [Per01] . En général, elle en interprète plus d'une car une description ignore certaines différences accessoires entre structures de démonstration.

Réciproquement, toute structure de démonstration peut être interprétée sous forme d'une description syntaxique à une condition près qui est que la première formule atomique qui précède la conclusion générale de la structure sur un chemin de parcours soit déjà connectée par un lien axiome. Cela provient de la contrainte 4) dans la définition d'une description syntaxique. Nous verrons à l'aide d'un exemple que cette condition n'est pas gênante si on intègre la ponctuation et notamment le point final aux mots de la phrase.

Les descriptions syntaxiques, telles qu'elles viennent d'être définies, vont constituer les objets syntaxiques de base des *grammaires d'interaction primitives*. Une grammaire d'interaction primitive est définie à partir d'un ensemble de catégories syntaxiques de base *Cat* par son lexique qui associe à chaque mot de la langue un ensemble de descriptions syntaxiques. Pour chaque description, un nœud est distingué comme l'*ancre* du mot auquel se rattache la description. Par rapport au modèle défini à la section 1.3, la notion d'entrée lexicale a été légèrement étendue. Elle peut maintenant représenter plus que l'arbre syntaxique d'une formule d'IILL : elle peut représenter une structure de démonstration où certains liens axiomes ont déjà été posés ; dans la description correspondante, cela se traduit par le fait que certains nœuds sont neutres. Cette extension de la notion de lexique est déjà présente chez A. Joshi et S. Kulick [JK97] de même que chez A. Lecomte et C. Retoré [LR98]. Les entrées lexicales se présentent sous la forme de morceaux de démonstration et le processus d'analyse consiste à composer ces différents morceaux pour former une démonstration complète. Les premiers utilisent les arbres de déduction naturelle du calcul de Lambek. Les seconds se placent dans le cadre des réseaux de démonstration d'une variante de la logique linéaire, le calcul ordonné.

Pour illustrer ce qu'est une grammaire d'interaction primitive, reprenons la phrase *« la personne que Jean voit aujourd'hui travaille »* et supposons qu'une

grammaire d'interaction primitive fournisse pour les mots de la phrase les entrées lexicales présentées par la figure 1.10. La description associée à chaque mot est une traduction de l'arbre syntaxique correspondant que l'on trouve sur la figure 1.6. S'y ajoute une ancre, représentée par un cercle plein, et l'ordre des syntagmes est maintenant exprimé par des relations binaires entre les nœuds, représentées par des flèches sur la figure 1.10. Comme nous l'avions déjà annoncé, le point final est considéré comme un mot de la phrase avec son entrée lexicale propre.

La réunion des descriptions syntaxiques fournies par le lexique pour les différents mots de la phrase, à laquelle il faut ajouter des relations de préséance entre les ancres pour exprimer l'ordre des mots dans la phrase, va constituer ce que nous appellerons la *spécification syntaxique* de la phrase et qui est présentée à la figure 1.10.

L'analyse syntaxique va consister à trouver un arbre syntaxique qui soit un modèle de cette spécification au sens de la théorie des modèles restreinte aux arbres ordonnés[5], un modèle respectant certaines conditions :

neutralité : tout nœud du modèle interprète soit un nœud de la description qui est neutre, soit deux nœuds duaux (avec des étiquettes opposées) ;

minimalité : toute relation de parenté du modèle interprète une relation de parenté de la description.

Ce modèle va être construit pas à pas en spécifiant de plus en plus la description de départ à l'aide de neutralisations élémentaires. Pour notre exemple, compte tenu des contraintes, nous n'avons qu'un modèle possible qui est obtenu à la suite de 8 neutralisations élémentaires et qui est l'arbre syntaxique présenté à la figure 1.11

Si les grammaires d'interaction primitives sont strictement équivalentes aux grammaires définies directement à partir des réseaux de démonstration intuitionnistes étiquetés de la section 1.3, elles présentent l'avantage d'une meilleure lisibilité car elles sont débarrassées de tous les aspects accessoires des réseaux de démonstration qui n'ont aucune pertinence linguistique. Nous verrons dans le chapitre suivant comment certaines de leurs limites vont pouvoir être dépassées dans le modèle plus élaboré de ce que nous appellerons les *grammaires d'interaction* tout court. Un formalisme qui est très voisin des grammaires d'interaction primitives, appelé LDG pour *Logical Description Grammar*, a été conçu indépendamment par R. Muskens à partir de l'expression des TAG sous forme de descriptions d'arbres [Mus01].

[5]Un arbre ordonné est un arbre où, pour chaque nœud, est défini un ordre partiel entre ses fils et cet ordre est ensuite étendu aux descendants.

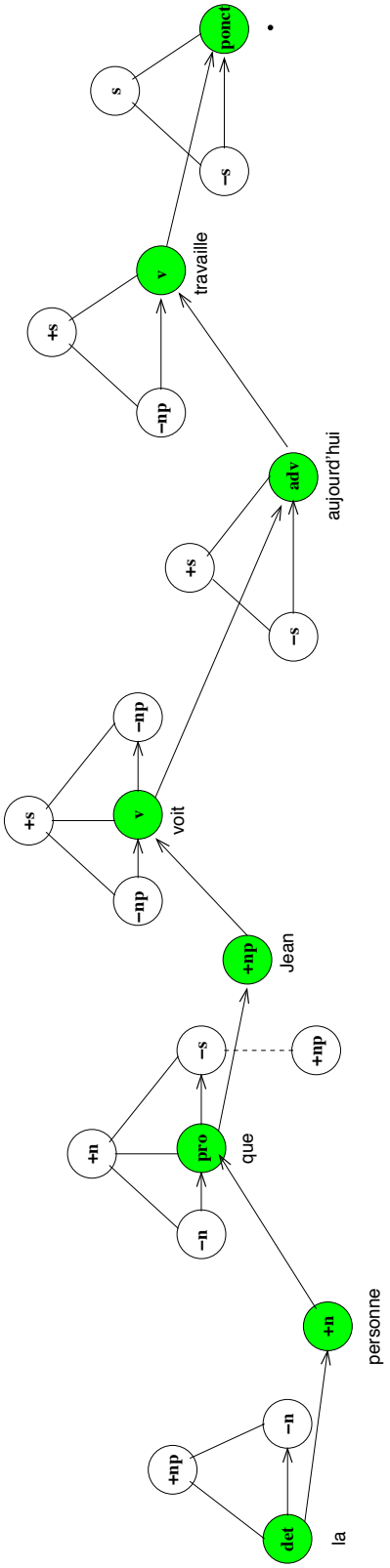


FIG. 1.10 – Description d'arbre exprimant la spécification syntaxique de la phrase *la personne que Jean voit aujourd'hui travaille*

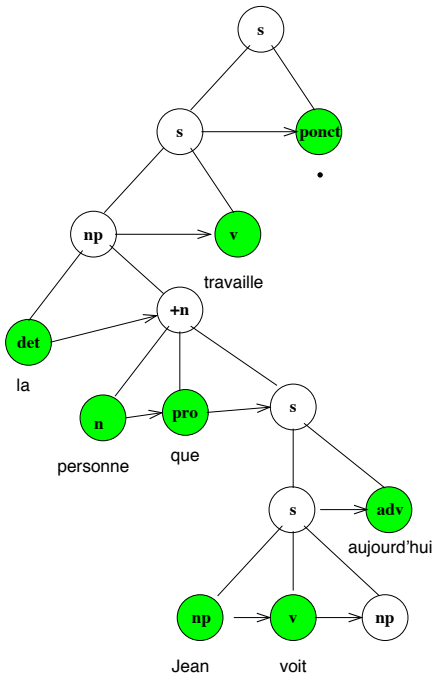


FIG. 1.11 – Arbre syntaxique de la phrase *la personne que Jean voit aujourd'hui travaille*

Chapitre 2

Le modèle syntaxique des grammaires d'interaction

Dans le chapitre précédent, nous avons montré comment en partant des réseaux de démonstration intuitionnistes nous en sommes arrivés au modèle syntaxique des grammaires d'interaction primitives.

Nous allons voir maintenant comment raffiner ce modèle pour exprimer certaines subtilités de la syntaxe des langues. Ce nouveau formalisme, que nous avons baptisé les *grammaires d'interaction*, va nous faire sortir du cadre de la logique linéaire dans lequel nous nous étions cantonnés jusqu'à maintenant. Nous verrons comment le définir mathématiquement et nous illustrerons son pouvoir expressif par un certain nombre d'exemples.

2.1 Une augmentation du pouvoir d'expression des grammaires d'interaction primitives

2.1.1 De la composition à la superposition d'arbres

Dans les grammaires algébriques, la composition syntaxique s'exprime par la substitution de la racine d'un arbre à une feuille d'un autre arbre, les deux étant étiquetées par un même non terminal. Dans les TAG, en plus de la substitution, la composition syntaxique peut s'effectuer par adjonction, ce qui consiste à insérer un arbre à l'intérieur d'un autre. Dans des formalismes plus sophistiqués qui dérivent des grammaires d'arbres adjoints comme les DTSG (D-Tree Substitution Grammars)[RVSW01], cette opération d'adjonction se complexifie pour devenir une opération d'entrelacement entre arbres.

Ce qui est important de noter dans tous ces formalismes, c'est que la composition syntaxique ne s'effectue jamais par une superposition d'arbres, si ce n'est une superposition minimale où on identifie la racine d'un sous-arbre avec un nœud d'un autre. Or, il est parfois très utile d'exprimer que deux mots destinés à être composés syntagmatiquement partagent un contexte syntaxique commun. Si on représente le contexte syntaxique de chaque mot par un arbre, la composition de ces deux mots va donner lieu à superposition partielle des deux arbres. Ceci n'est pas possible dans la plupart des formalismes basés sur les arbres, ce qui en limite grandement le pouvoir et la souplesse d'expression.

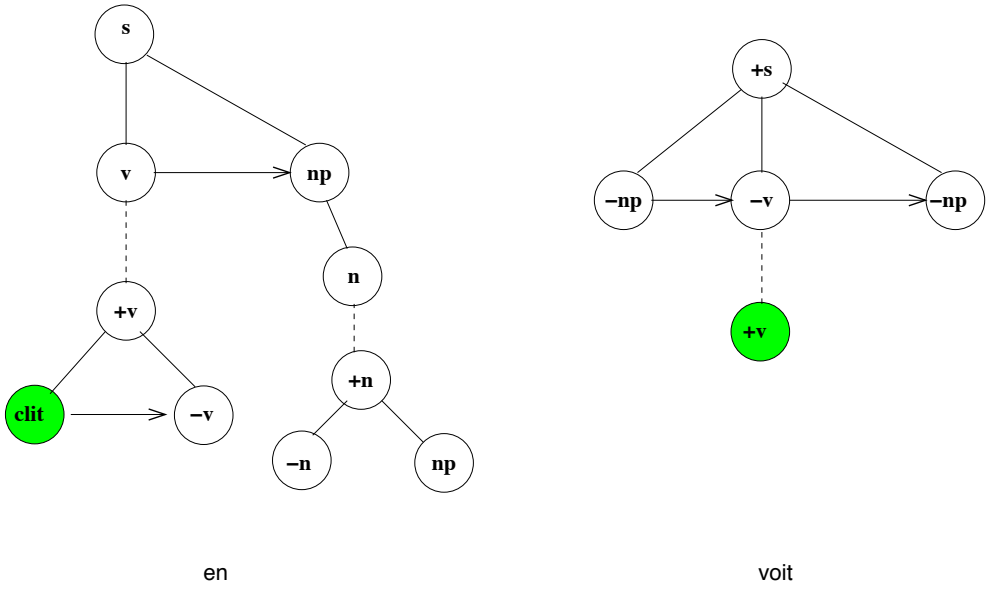


FIG. 2.1 – Entrées lexicales de *en* et *voit*

Prenons l'exemple du pronom personnel clitique *en* lorsqu'il joue le rôle de complément de nom de l'objet d'un verbe transitif comme dans la phrase « *Jean en voit la silhouette* ». Intéressons nous plus particulièrement à la composition syntaxique de *en* avec *voit*. Nous supposons qu'une grammaire d'interaction fournisse pour ces deux mots les entrées lexicales de la figure 2.1. L'entrée lexicale du verbe transitif *voit* est légèrement différente de celle qui apparaît dans la figure 1.10. Le nœud v a été dissocié en deux nœuds $+v$ et $-v$ pour exprimer la possibilité de modifier le verbe par des adverbes ou des clitiques.

L'entrée lexicale du pronom personnel clitique *en* comporte deux parties :

- La première exprime le fait que le pronom, en tant que clitique vient modifier un verbe ; le nœud $-v$ représente le verbe attendu et le nœud $+v$ représente le verbe cliticisé qui sera fourni en retour.
- La seconde exprime la fonction grammaticale de *en*. Le nœud *np* fils du nœud s représente l'objet du verbe, verbe dont la projection maximale est représentée par le nœud v. Cet objet est un syntagme nominal qui est nécessairement formé à partir d'un nom commun ; il n'est pas possible en effet de dire « * *il en voit Marie* ». Ce nom commun est modifié par le pronom *en* ; le nœud $-n$ représente le nom commun attendu et le nœud $+n$ représente le nœud fourni après modification par *en* qui donne lieu à une trace vide comme complément de nom, la feuille *np*. Il y a une relation de domination entre la projection maximale n du nom commun et le nœud $+n$ pour tenir compte du fait que le nom commun peut ensuite être modifié par des adjectifs.

Cherchons maintenant à composer *en* avec *voit*. Dans les grammaires d'interaction primitives, c'est l'opération de neutralisation élémentaire entre nœuds duaux qui permet de réaliser cette composition. Dans le cas qui nous intéresse, ce sont deux neutralisations élémentaires entre tous les nœuds $+v$ et $-v$ qui le

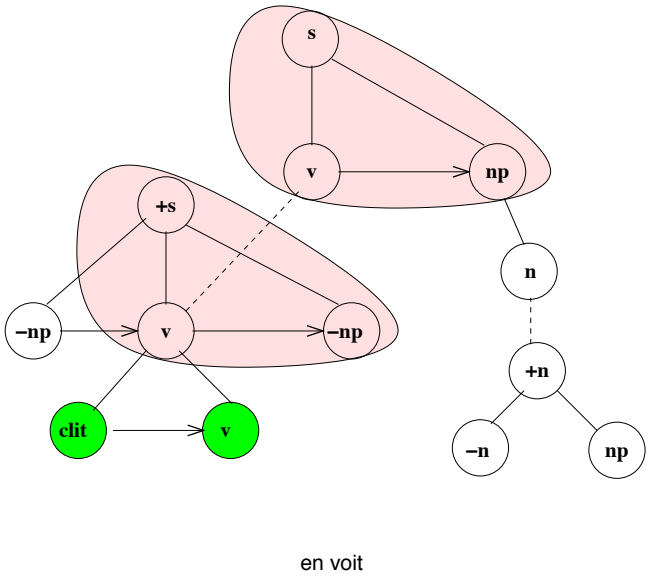


FIG. 2.2 – Composition syntaxique de *en* avec *voit* réalisée uniquement par des neutralisations élémentaires

font et il en résulte la description de la figure 2.2. Pour obtenir la description attendue, nous souhaiterions pouvoir superposer les parties d'arbres grisées mais le formalisme des grammaires d'interaction primitives ne le permet pas puisqu'il permet uniquement de superposer deux nœuds porteurs de catégories grammaticales opposées. Dit d'une autre façon, les nœuds d'un modèle ne peuvent interpréter que deux nœuds duaux d'une description ou un nœud neutre seul. Ils ne peuvent interpréter à la fois un nœud neutre et un autre nœud comme ce serait nécessaire dans le cas qui nous intéresse.

Ceci nous amène à assouplir le formalisme de la façon suivante :

> *Un nœud d'un modèle peut interpréter au maximum deux nœuds duaux mais un nombre quelconque de nœuds neutres.*

En termes opérationnels, cela signifie qu'il faut relâcher aussi l'opération élémentaire de neutralisation. Nous admettrons provisoirement une autre opération élémentaire que nous baptiserons *superposition élémentaire* et qui consiste à fusionner un nœud neutre avec un nœud quelconque de la même catégorie, le nœud résultant étant alors porteur de l'étiquette du second. Dans ces conditions, il est possible d'effectuer la superposition attendue à l'aide de 3 superpositions élémentaires et il en résulte la description de la figure 2.3.

Ce faisant, nous sortons du cadre logique d'IILL et même d'IMLL. Nous pourrions montrer qu'en utilisant les connecteurs additifs de la logique linéaire, nous sommes capables d'exprimer cette superposition de manière logique mais ce serait au prix d'une lourdeur dans la traduction. C'est pourquoi nous ne nous livrerons pas à ce genre d'exercice. Nous avons préféré définir directement un nouveau formalisme sans aucune référence à la logique linéaire. C'est ce que nous présenterons à la section 2.2.

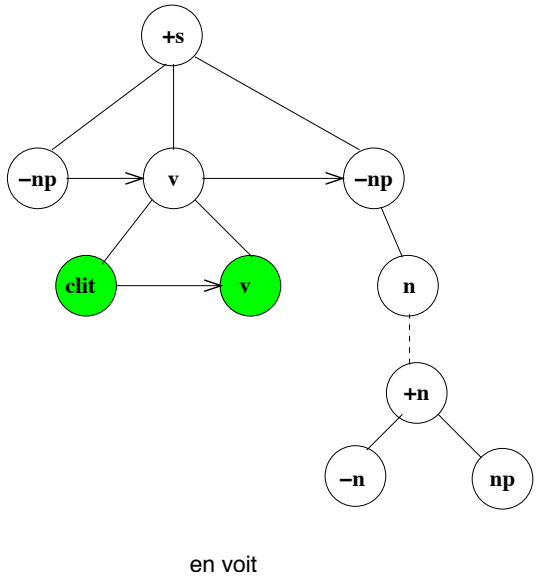


FIG. 2.3 – Composition syntaxique de *en* avec *voit* réalisée par des neutralisations et superpositions élémentaires

2.1.2 Contraintes sur les relations de domination et de parenté

Le fait de relâcher le formalisme en permettant la superposition de nœuds neutres a son pendant qui est la sur-génération. Il est donc nécessaire d'introduire de nouvelles contraintes qui vont permettre de contrôler les nouvelles potentialités du formalisme.

Dépendances non bornées et contraintes sur les relations de domination

Les dépendances syntaxiques non bornées posent des sérieux problèmes de représentation et de traitement aux différents formalismes linguistiques. Dans les TAG [AR01], l'adjonction autorise le principe de localité étendue et par là même l'expression de dépendances non bornées. Dans HPSG [PS94], c'est l'utilisation de traits spécifiques qualifiés de non locaux et leur propagation de proche en proche qui le permet. Dans les grammaires d'interaction primitives, c'est la relation de domination large entre nœuds syntaxiques qui rend possible la représentation des dépendances syntaxiques non bornées. Alors que la technique utilisée dans HPSG contredit le principe de localité étendue, la relation de domination large suit ce principe qui est présenté comme un atout des TAG.

Prenons l'exemple de l'extraction et en particulier l'extraction des propositions relatives[6]. Considérons le pronom relatif objet *que*. Le syntagme nominal qui constitue son antécédent peut être considéré comme extrait de la proposition que le pronom relatif introduit. Dans la relative, il remplit la fonction d'objet mais cet objet n'est pas forcément celui du verbe principal de la re-

[6]Les extractions des interrogatives se traitent de façon similaire.

lative : si la relative contient une suite de complétives imbriquées, l'objet est celui du verbe de la complétive qui se trouve au niveau le plus profond. Soit la phrase *« la personne que Marie pense que Jean voit aujourd'hui travaille. »*. Avec l'entrée lexicale du pronom relatif *que* qui est présentée à la figure 1.10, il est possible d'analyser cette phrase. On obtient alors l'arbre de la figure 2.4.

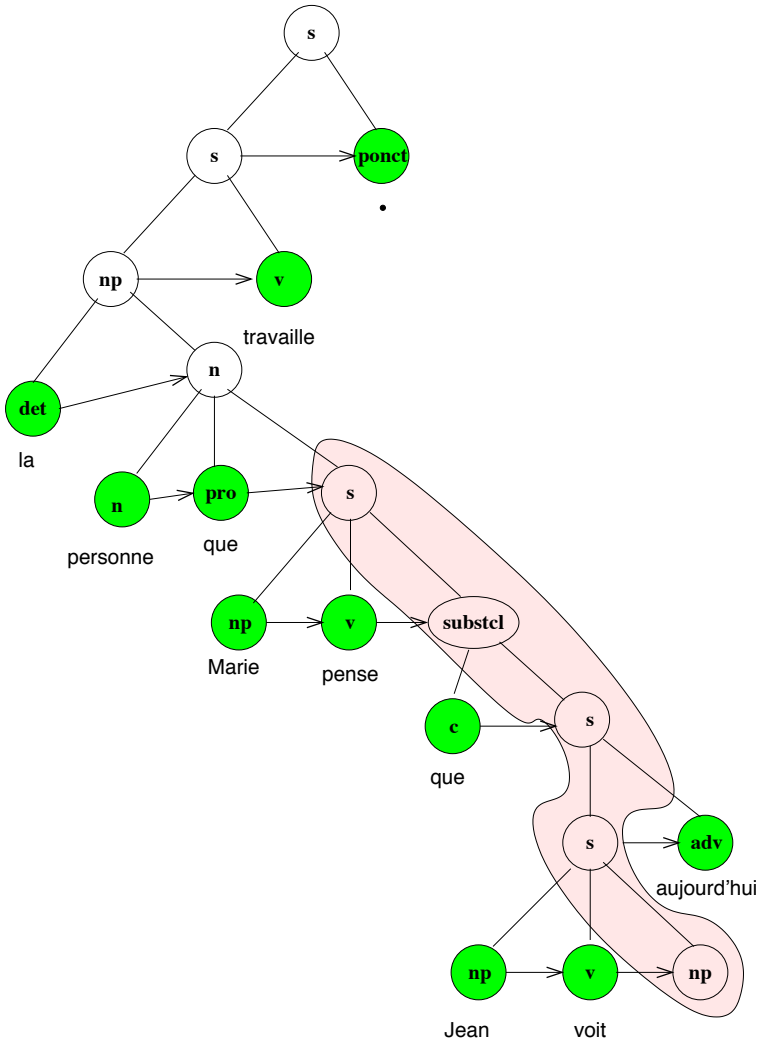


FIG. 2.4 – Arbre d'analyse de la phrase *« la personne que Marie pense que Jean voit aujourd'hui travaille. »*

Dans l'entrée lexicale de *que*, telle qu'elle est représentée à la figure 1.10, le nœud $-s$ représentant la proposition relative attendue sans le pronom qui l'introduit, domine la trace de l'objet extrait qui est représentée par le nœud $+np$. Le fait que la profondeur de la domination soit laissée sous-spécifiée autorise à ce que cette trace soit l'objet d'un verbe d'une complétive imbriquée à un niveau quelconque. Pour ce qui concerne notre exemple, la relation de domination de la relative sur la trace est réalisée par le chemin de longueur 4 $s - substcl - s - s - np$, indiqué en grisé sur la figure 2.4.

C'est parce qu'il n'y a pas de contrainte quant à la profondeur à laquelle

se situe la trace de l'objet extrait que l'on parle de dépendance non bornée. Ce n'est pas pour autant que cet objet peut venir de n'importe quel syntagme présent dans la relative. Il y a un certain nombre de barrières qui interdisent l'extraction. Considérons la phrase incorrecte * *« la personne que Jean qui voit aujourd'hui arrive travaille. »*. Avec le même lexique que précédemment auquel on ajoute une entrée pour le pronom relatif *qui* et pour le verbe intransitif *arrive* on peut pourtant analyser cette phrase et on trouve l'arbre syntaxique de la figure 2.5.

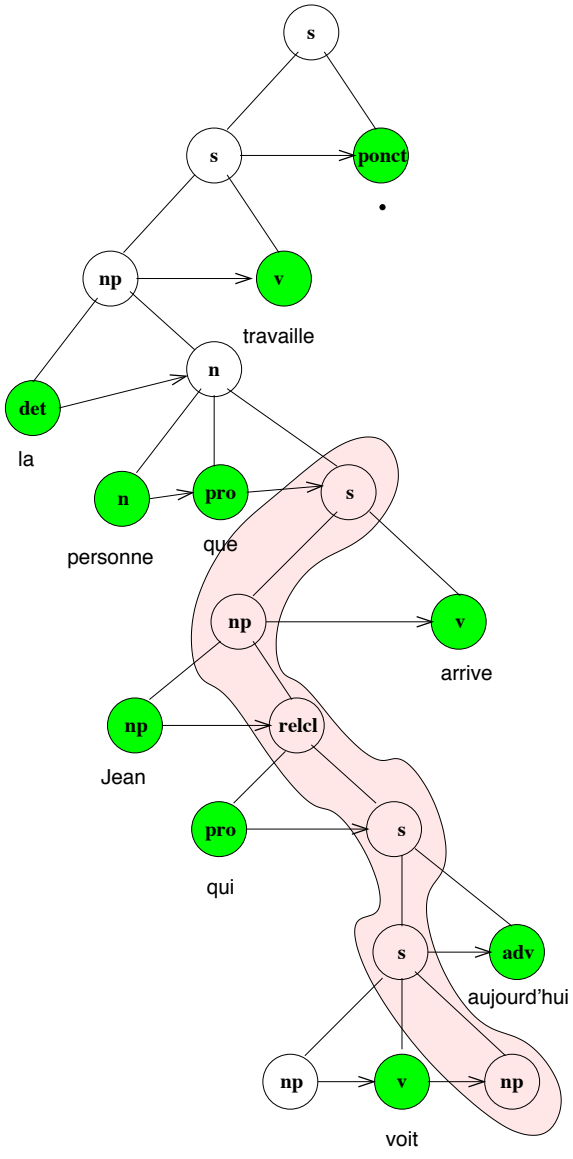


FIG. 2.5 – Arbre d'analyse de la phrase incorrecte *« * la personne que Jean qui voit aujourd'hui arrive travaille. »*

La relation de domination entre la relative introduite par *que* et la trace de l'objet déplacé est réalisée par le chemin de longueur 5 $s-np-relcl-s-s-np$, indiqué en grisé sur la figure 2.5. L'incorrection vient du fait que la relative

introduite par *qui* constitue une barrière à l'extraction.

Comment enrichir le formalisme pour exprimer ces barrières ? Il suffit d'ajouter des contraintes sur les relations de domination. Provisoirement, nous définirons une contrainte comme un ensemble de catégories grammaticales $\{c_1, \ldots, c_n\}$ attaché à une relation de domination particulière où un nœud N_1 domine un nœud N_2 et nous lui donnerons la signification suivante :

> *Tout nœud qui domine strictement N_2 et qui est dominé au sens large par N_1 doit être étiqueté par une des catégories grammaticales de l'ensemble $\{c_1, \ldots, c_n\}$.*

Si N_1 est confondu avec N_2, cette contrainte ne joue pas et, sinon, elle s'applique à tous les nœuds strictement au-dessus de N_2 jusqu'à N_1 compris. Dans notre exemple, dans l'entrée lexicale de *que*, telle qu'elle est présentée sur la figure 1.10, il suffit d'ajouter la contrainte $\{s,\ substcl\}$ à la relation de domination entre le nœud $-s$ qui représente la proposition relative sans le pronom et le nœud $+np$ qui représente la trace de l'objet. Ainsi, tout nœud qui dominera strictement la trace et qui sera dominé par la relative devra être soit une proposition (s), soit une complétive (*substcl*). Cela interdira en particulier l'analyse de la figure 2.5 à cause des nœuds *np* et *relcl* dans la chaîne de domination de la trace.

On ne peut pas éviter de rapprocher cette notion de domination contrainte du concept d'incertitude fonctionnelle de LFG [KZ89]. Pour représenter les dépendances syntaxiques à longue distance, R. Kaplan et A. Zaenen ont introduit des équations fonctionnelles qui identifient l'élément extrait porteur d'une fonction du discours (TOPIC ou FOCUS) avec celui remplissant une fonction grammaticale, OBJ par exemple, plus loin dans la phrase. Comme cette fonction peut être réalisée à travers une série de complétives et d'infinitives dont le nombre est indéfini, le chemin constitué par la suite des dépendances fonctionnelles allant de la tête dont dépend directement l'élément topicalisé jusqu'à la dépendance OBJ est représenté par une expression régulière : COMP* si on désigne par COMP la fonction complétive ou infinitive objet.

De façon plus précise, dans une grammaire de type LFG, une relative introduite par le pronom objet *que* pourrait être représentée à l'aide de la règle algébrique et des équations fonctionnelles suivantes :

$$\begin{array}{lccc} S' & \longrightarrow & que & S \\ & & \uparrow TOPIC = \downarrow & \uparrow = \downarrow \\ & & (\uparrow COMP^\star OBJ) = \downarrow & \end{array}$$

Dans les grammaires d'interaction, la sous-spécification est exprimée au niveau de la structure syntagmatique à l'aide d'une relation de domination large. Dans LFG, elle est exprimée au niveau des fonctions syntaxiques à l'aide d'une équation d'incertitude fonctionnelle : $(\uparrow COMP^\star OBJ) = \downarrow$. Les contraintes d'extraction sont exprimées ici en termes fonctionnels à l'aide de COMP, alors que dans les grammaires d'interaction, elles sont exprimées en termes de catégories grammaticales s et *substcl*.

Dans les grammaires d'interaction, les relations de domination large avec contrainte ont une utilisation plus étendue que les équations d'incertitude fonctionnelle de LFG et ne servent pas seulement à représenter des dépendances

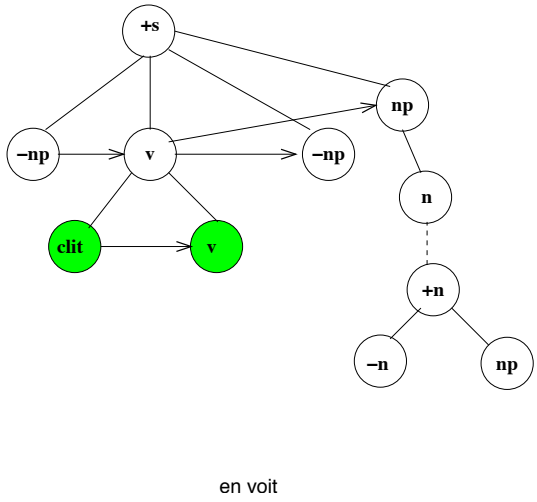


FIG. 2.6 – Composition syntaxique de *en* avec *voit* réalisée par des neutralisations élémentaires suivies d'une simplification

syntaxiques non bornées. Par exemple, dans l'entrée lexicale du pronom personnel *en*, telle qu'elle apparaît à la figure 2.1, il y a une relation de domination large entre le nœud n représentant la projection maximale du nom commun tête du complément d'objet et le nœud $+n$ représentant le nom commun modifié par *en*. Entre les deux, il ne peut y avoir que des nœuds de catégorie grammaticale n provenant de modifieurs du nom commun donc on peut ajouter la contrainte $\{n\}$ sur la relation de domination large.

Plus intéressant par les conséquences que cela va avoir, il y a une relation de domination large entre le nœud v représentant la projection maximale du verbe et le nœud $+v$ représentant le verbe cliticisé mais on sait qu'entre les deux il ne peut y avoir que des nœuds de la catégorie grammaticale v donc on peut ajouter la contrainte $\{v\}$ sur la relation de domination large. Ceci a pour effet de forcer une superposition à la suite de la composition syntaxique qui produit la description de la figure 2.3. En effet, étant donné la contrainte $\{v\}$ sur la relation de domination, la source de cette relation ne peut pas dominer le père de sa cible, le nœud $+s$, donc elle est nécessairement confondue avec sa cible. Cette fusion entraîne ensuite la fusion des deux pères, étant donné l'unicité du père de chaque nœud dans un arbre. En définitive, on obtient la description de la figure 2.6. Une partie de la superposition souhaitée est donc obtenue automatiquement comme conséquence de la contrainte de domination. Il reste encore à superposer les deux nœuds *np* qui représentent tous les deux l'objet du verbe. Une telle superposition peut aussi être obtenue automatiquement par une contrainte qui clôt l'ensemble des fils d'un nœud donné. C'est ce que nous allons voir maintenant.

Contrainte de clôture de l'ensemble des fils d'un nœud syntaxique

Souvent, le nombre et la catégorie des constituants immédiats d'un syntagme sont connus à l'avance. Par exemple, on sait que, dans sa configuration

canonique, une phrase simple en français dont la tête est un verbe transitif est formée de 3 constituants immédiats qui sont le sujet, le verbe et l'objet. En termes d'arbre syntaxique, cela signifie que le nombre et l'étiquette des fils de certains nœuds sont connus d'avance. En permettant la superposition de nœuds neutres, nous avons ouvert la possibilité d'ajouter un nombre indéterminé de fils à un nœud donné. Il est donc nécessaire d'ajouter au formalisme une contrainte pour limiter cette possibilité. Cette contrainte consiste à figer l'ensemble $\{N_1, \ldots, N_p\}$ des fils d'un nœud N, cet ensemble pouvant être vide et ses éléments étant tous distincts. La conséquence en est que toute relation de parenté qui est introduite entre le nœud N et un nœud M implique que M est un élément de $\{N_1, \ldots, N_p\}$.

Reprenons l'exemple de *« en voit »*. Si dans l'entrée lexicale du verbe transitif *voit*, telle qu'elle est présentée sur la figure 2.1, nous ajoutons la contrainte que l'ensemble des fils du nœud $+s$ est uniquement formé des deux nœuds $-np$ et du nœud $-v$, cela aura comme conséquence que la superposition souhaitée qui avait été partiellement réalisée à la figure 2.6 va pouvoir être achevée par la superposition élémentaire des deux nœuds de type np fils du nœud $+s$. Nous retrouverons alors exactement ce que nous souhaitions, c'est-à-dire la description de la figure 2.3.

2.1.3 Des syntagmes polarisés aux traits morpho-syntaxiques polarisés

La façon classique de représenter les propriétés morpho-syntaxiques des différents syntagmes est d'utiliser des structures de traits. Pour l'instant, dans les grammaires d'interaction primitives, les nœuds qui représentent les syntagmes sont seulement étiquetés par leur catégorie grammaticale qui est polarisée mais rien n'empêche de les étiqueter par des structures de traits.

Jusqu'à présent aussi, les polarités portaient sur les nœuds. Si nous étiquetons ces nœuds par des structures de traits, pourquoi ne pas faire porter les polarités sur les traits eux-mêmes ? Les traits se comportant comme des ressources attendues seraient négatifs, ceux se comportant comme des ressources disponibles seraient positifs et les autres exprimant des propriétés ne se comportant pas comme des ressources consommables seraient neutres. Cela permettrait d'avoir des nœuds porteurs à la fois de traits positifs et négatifs et donc susceptibles d'interagir de façon plus subtile entre eux. Illustrons ceci par deux exemples : l'inversion du sujet dans les relatives introduites par *que* et la négation.

Inversion du sujet dans les relatives introduites par *que*

La phrase *« la personne que voit Jean travaille. »* présente ce phénomène d'inversion dans la relative *« que voit Jean »*. Cette inversion est possible car il n'y a pas d'ambiguïté sur le fait que *Jean* puisse être l'objet de *voit* : cet objet est déjà déterminé par le pronom relatif *que*, ce qui permet de relâcher l'ordre sujet-verbe. Il est difficile de rendre compte de ce phénomène avec les grammaires d'interaction primitives sans sur-générer ou sans multiplier les syntagmes vides.

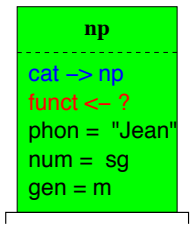


FIG. 2.7 – Entrée lexicale pour le nom propre *Jean*

Voyons comment faire en descendant les polarités au niveau des traits. La figure 2.7 présente une entrée lexicale possible pour le nom propre *Jean* utilisant des traits polarisés. Les nœuds syntaxiques sont maintenant représentés par des rectangles qui ont comme entête le nom du nœud et comme corps la liste des traits attachés au nœud. Les nœuds qui sont des ancres sont représentés par des rectangles pleins. Les traits morpho-syntaxiques positifs, négatifs et neutres sont représentés respectivement à l'aide des symboles $\rightarrow$,$\leftarrow$ et $=$. Les valeurs des traits peuvent être indéterminées, auquel cas elles sont représentées par le symbole « ? ». Dans notre exemple, $cat \rightarrow np$ signifie que le nœud est capable de fournir un syntagme nominal et $funct \leftarrow ?$ signifie qu'il attend de recevoir une fonction syntaxique qui n'est pas encore déterminée. Les contraintes signifiant que l'ensemble des fils d'un nœud est clos sont représentées par des crochets tournés vers le bas placés à la base des rectangles correspondants. Dans notre exemple, la contrainte signifie que le nœud ne peut pas recevoir de fils.

Voyons maintenant comment représenter l'inversion du sujet dans les relatives à l'aide de traits polarisés. Pour cela, il est nécessaire de considérer deux entrées lexicales pour un verbe transitif : la première représenterait la construction canonique sujet-verbe-objet et la seconde représenterait une construction où l'objet est une trace avec une forme phonologique vide et où donc le sujet est libre dans sa position par rapport au verbe. La figure 2.8 présente ces deux entrées lexicales. Les valeurs des traits sont parfois partagées, auquel cas elles sont précédées d'un indice. Les contraintes sur les relations de domination sont indiquées directement sur les traits en pointillés représentant ces relations sous la même forme que les traits étiquetant les nœuds.

La description représentant la construction canonique sujet-verbe-objet signifie que le verbe attend deux syntagmes nominaux : l'un à sa gauche auquel il va fournir la fonction de sujet et l'autre à sa droite auquel il va fournir la fonction d'objet. La description représentant la construction libre signifie aussi que le verbe attend deux syntagmes nominaux mais l'ordre est maintenant libre. En plus si le verbe va fournir la fonction objet à celui qui a une forme phonologique vide, ce n'est plus le cas pour le sujet. La fonction sera fourni dans le processus de composition syntaxique avec un autre mot.

Nous allons maintenant proposer une entrée lexicale du pronom relatif *que* qui tienne compte de ces deux nouvelles entrées. Elle se présente comme l'indique la figure 2.9. Par rapport à l'entrée lexicale qui était proposée à la figure 1.10, la principale nouveauté tient dans les nœuds *subj2* et *obj2*. Le second va fournir un syntagme nominal vide et recevoir la fonction objet d'un verbe tran-

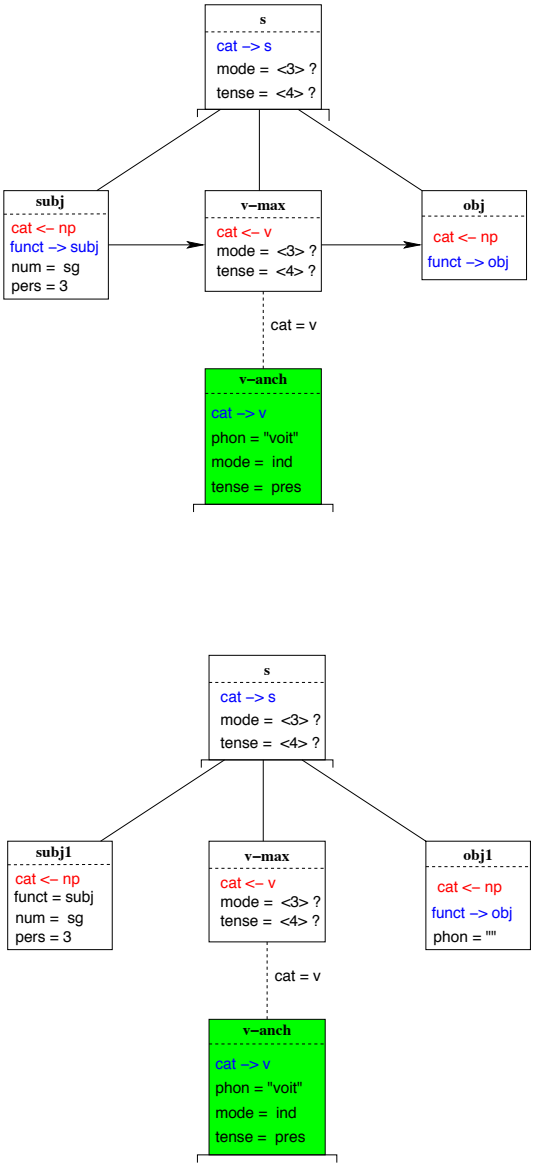


FIG. 2.8 – Deux entrées lexicales pour le verbe transitif *voit*

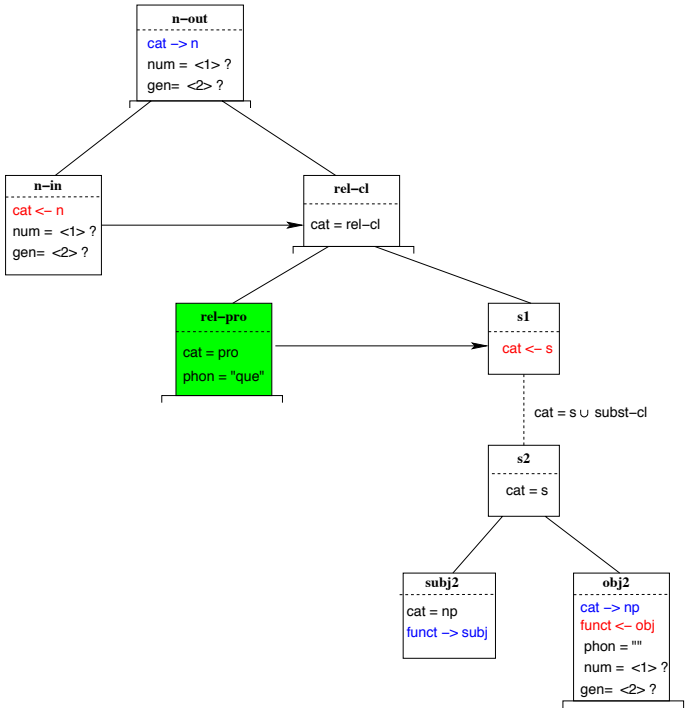


FIG. 2.9 – Entrée lexicale pour le pronom relatif *que*

sitif alors que le premier va fournir la fonction sujet à un éventuel syntagme en attente de recevoir cette fonction, ceci quelque soit la position de ce syntagme par rapport au verbe. Le père *s2* des deux nœuds *subj2* et *obj2* est aussi explicite et c'est lui maintenant que domine *s1*. La valeur de la contrainte sur la relation de domination $s \cup subst\text{-}cl$ doit être comprise comme une disjonction ; elle correspond exactement à celle qui était notée *{s, subst-cl}* dans la section 2.1.2.

Maintenant, si nous composons dans l'ordre *que* avec *voit* et *Jean*, nous obtenons la description de la figure 2.10. La proposition relative qui en résulte, *« que voit Jean »*, apparaît bien comme un modifieur de nom commun, compte tenu des deux traits non neutres seulement qui restent. Avec exactement les mêmes entrées lexicales, on aurait pu composer tout aussi bien l'expression *que Jean voit* [7]..

La négation

La négation en français pose des problèmes syntaxiques du fait qu'elle s'exprime en général sous forme de couples de deux termes dont la corrélation est réalisée plus ou moins souplement par la syntaxe. Prenons le couple *ne... aucun* où justement la corrélation entre les deux termes est très souple. Si *ne*

[7] M. Steedman a justement fait remarqué que cela ne règle pas tous les problèmes et que l'on évite pas la sur-génération de * *un homme que mon oncle m'a dit que voit Jean*. C'est pourquoi il semble difficile de se passer de deux entrées lexicales pour le pronom relatif *que* : la première ne permet pas l'inversion du sujet mais elle autorise la dépendance non bornée de l'objet et pour la seconde c'est le contraire.

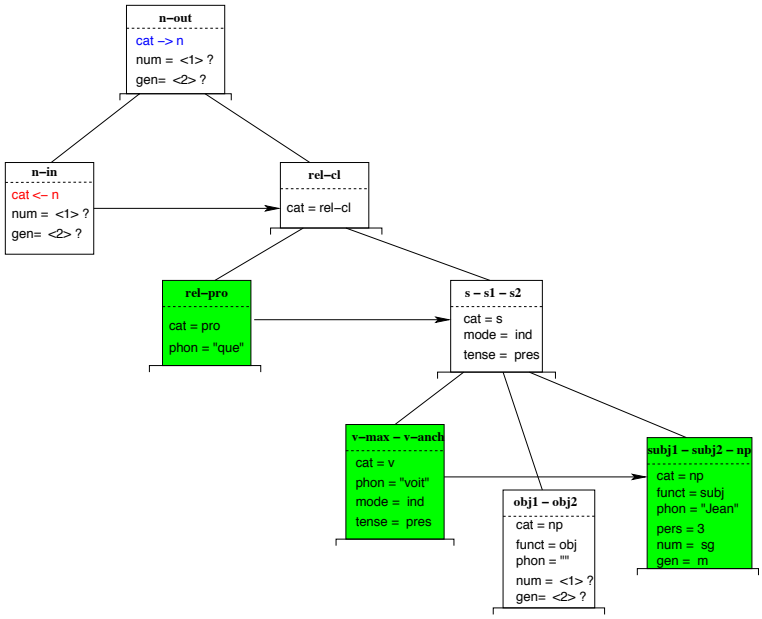


FIG. 2.10 – Description résultant de la composition syntaxique de *« que voit Jean »*

se place juste devant le verbe auquel il s'applique, *aucun* est un déterminant qui peut tout aussi bien s'appliquer au sujet qu'à n'importe quel complément du verbe et même à un complément de nom imbriqué plus ou moins profondément dans un constituant immédiat de la phrase. Voici quelques exemples d'utilisation de cette négation.

aucune personne ne travaille
Jean ne voit aucune personne
Jean ne voit Marie dans aucune chambre
Jean ne voit Marie sur le pas de la porte d'aucune chambre

Avec une grammaire du type HPSG ou TAG, on peut exprimer la corrélation entre *ne* et *aucun* à l'aide d'un trait *neg* à valeur booléenne mais cela entraîne des lourdeurs pour gérer sa propagation et cela n'évite pas par exemple la présence de deux occurrences de *aucun* qui est agrammaticale.

L'utilisation d'un trait polarisé *neg* va nous permettre de représenter cette corrélation de façon souple et économique. La particule *ne* va fournir un trait positif *neg* à la projection maximale du verbe qui suit et le déterminant *aucun* va lui pourvoir la projection maximale du verbe qui est la tête de la proposition dans laquelle il s'insère d'un trait négatif *neg* avec la même valeur. De cette façon sera réalisée la corrélation entre exactement un *ne* et un *aucun*. La figure 2.11 nous montre précisément comment ce mécanisme peut se réaliser à travers les entrées lexicales de *ne* et de *aucun*. Dans l'entrée lexicale de *aucun*, il y a deux relations de domination large.

La première relation entre les nœuds *s2* et *v-max2* avec la contrainte *cat* $= s$ exprime qu'il puisse y avoir un nombre indéterminé de nœuds s entre les deux du fait de modifications possibles de la phrase par des adverbes ou des

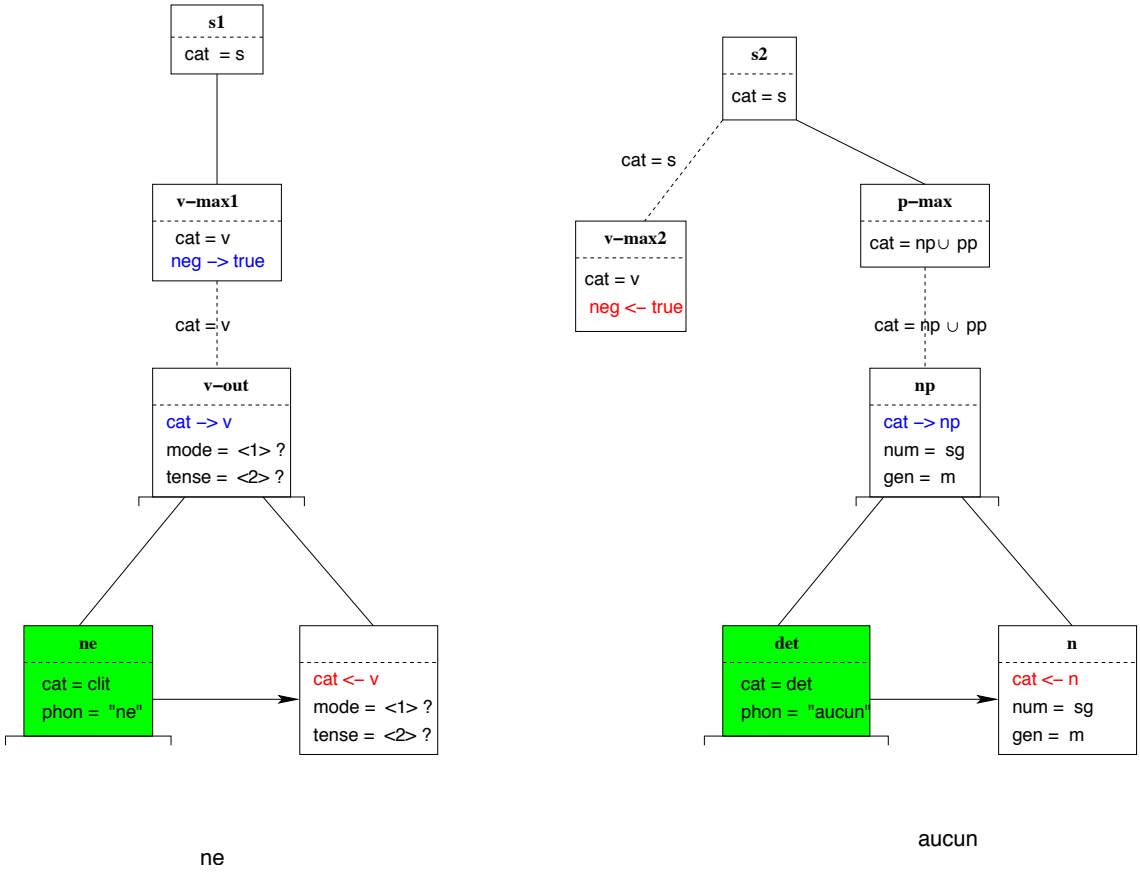


FIG. 2.11 – Entrées lexicales pour *ne* et *aucun*

compléments circonstanciels, *aucun* pouvant être dans un complément circonstanciel.

La seconde relation entre les nœuds *p-max* et *np* avec la contrainte $cat = np \cup pp$ exprime que *aucun* est dans un syntagme nominal qui peut être enfoui plus ou moins profondément dans le syntagme propositionnel ou nominal qui est complément du verbe sur lequel porte la négation.

2.2 La définition formelle des grammaires d'interaction

Nous venons de montrer comment enrichir les grammaires d'interaction primitives pour exprimer certains phénomènes syntaxiques. Il s'agit maintenant de donner une description complète et formelle du modèle qui en résulte et que nous avons baptisé *grammaires d'interaction* tout court.

f	$\mathcal{V}_\mathbf{f}$
cat	s, np, n, rel-cl, fin-cl, inf-cl, det, pro
funct	subj, obj, a-obj, de-obj, noun-compl
mode	ind, subj, inf, past-p, pres-p
tense	pres, imperf, fut
gen	m, f
num	sg, pl
pers	1, 2, 3

TAB. 2.1 – Exemples de valeurs de traits morpho-syntaxiques

2.2.1 Les descriptions syntaxiques comme descriptions d'arbres polarisées

Le formalisme des grammaires d'interaction s'appuie sur deux notions clés : celle de *description d'arbre* et celle de *polarité*. Si la première est familière notamment à ceux qui connaissent les TAG [VS92, RVSW01], ce n'est pas le cas de la seconde. Celle-ci a été utilisée par D. Duchier and S. Thater pour développer un modèle syntaxique très proche des grammaires d'interaction primitives, c'est-à-dire où ce sont les syntagmes et non les traits qui sont polarisés [DT99]. Dans le modèle final des grammaires d'interaction, ce sont les traits qui sont polarisés et non les syntagmes et cette polarisation va jouer un rôle central dans le mécanisme de composition syntaxique. Commençons par la définir.

Les traits polarisés

Soit $\mathcal{F}$ un ensemble de noms de traits morpho-syntaxiques. Nous utiliserons par exemple pour la suite l'ensemble $\mathcal{F}$ = *{cat, funct, phon, mode, tense, gen, num, pers}*, où les éléments sont des abréviations pour les expressions en anglais *grammatical category, syntactic function, phonological form, mode, tense, gender, number, person.*

Chaque trait f de $\mathcal{F}$ est associé à l'ensemble $\mathcal{V}_f$ de ses valeurs atomiques. La table 2.1 nous donne quelques exemples de ces valeurs. Tous les traits ont leur domaine de valeurs atomiques qui est fini sauf le trait *phon* qui représente la forme phonologique du syntagme.

A partir de l'ensemble $\mathcal{V}_f$ des valeurs atomiques du trait f, nous allons pouvoir construire le domaine $\mathcal{D}_f$ des valeurs de f.

Définition 2.2.1. *Le domaine $\mathcal{D}_f$ des valeurs de f est l'ensemble des disjonctions finies $v_1 \cup \cdots \cup v_n$ de valeurs atomiques $v_1, \ldots, v_n$ de $\mathcal{V}_f$. Cette disjonction peut être vide et dans ce cas, la valeur est la valeur incohérente $\emptyset$.*

Le domaine $\mathcal{D}_f$ peut être structuré en treillis distributif avec comme opération de borne supérieure, la disjonction $\cup$. Le treillis étant distributif, il suffit de définir l'opérateur de borne inférieure pour deux éléments distincts v_1 et v_2 de $\mathcal{V}_f$. On posera alors : $v_1 \cap v_2 = \emptyset$. La valeur incohérente $\emptyset$ constitue donc le minimum du treillis. Si le domaine des valeurs atomiques $\mathcal{V}_f$ est fini, le treillis

possède un maximum qui est la valeur $\bigcup_{v \in \mathcal{V}_f} v$. Cette valeur, qu'on appellera la *valeur indéterminée*, sera notée « f ».

Certaines disjonctions de valeurs ont plus de pertinence linguistique que d'autres et vont être utilisées très souvent. Pour les nommer, nous utiliserons des abréviations. Prenons par exemple le trait *cat* et la disjonction *fin-cl* ∪ *inf-cl* qui signifie *complétive à un temps conjugué ou infinitive*. Nous pouvons utiliser l'abréviation *subst-cl* pour nommer cette valeur qui signifie *complétive* au sens large. Maintenant, la valeur *np* ∪ *subst-cl* peut être aussi remplacée par l'abréviation *subst* qui représente la valeur *substantif*.

Définition 2.2.2. *Un* trait polarisé *est un triplet* (f, p, v) *où* f *est un nom de trait et* v *une valeur de* $\mathcal{D}_f$. *La composante* p *représente une polarité qui peut prendre quatre valeurs :* positif, négatif, neutre *et* annulé *représentées respectivement par les symboles* $\rightarrow$, $\leftarrow$, $=$ *et* $\leftrightarrow$.

On simplifie alors l'écriture des quatre types de traits possibles $(f, \rightarrow, v)$, $(f, \leftarrow, v)$, $(f, =, v)$, $(f, \leftrightarrow, v)$ ainsi :$f \rightarrow v$, $f \leftarrow v$, $f = v$, $f \leftrightarrow v$. La polarité $\leftrightarrow$, qui apparaît ici pour la première fois, trouve sa raison d'être dans la nécessité de distinguer un trait qui provient de la neutralisation de deux traits de polarités opposée d'un trait qui est initialement neutre. Un trait de polarité $\leftrightarrow$ ne pourra plus s'unifier avec un trait positif ou négatif, ce qui n'est pas le cas d'un trait de polarité $=$.

Comme pour les traits non polarisés, les traits polarisés sont regroupés en structures de traits polarisés qui vont être associées à des syntagmes.

Définition 2.2.3. *Une* structure de traits polarisés *est une fonction partielle qui associe à chaque nom de trait* f *de* $\mathcal{F}$ *au plus un trait polarisé* (f, p, v).

Comme l'ensemble $\mathcal{F}$ des noms de traits est fini, nous pourrons aussi définir une structure de traits par son extension $\{(f_1, p_1, v_1), \ldots, (f_n, p_n, v_n)\}$.

Nous souhaitons pouvoir exprimer que plusieurs traits partagent la même valeur. Pour le formaliser, nous allons utiliser une notion d'environnement. A chaque nom de trait f, nous allons associer un ensemble dénombrable de noms de variables d'environnement $\mathcal{E}_f$.

Nous aurons besoin dans le processus d'analyse syntaxique d'identifier des variables d'environnement. Pour définir cette opération de façon rigoureuse, nous introduirons une relation d'équivalence entre noms de variables et ce que nous appellerons variable sera non pas un nom de variable mais une classe d'équivalence, c'est-à-dire un ensemble de noms. Ainsi, identifier deux variables reviendra à étendre la relation d'équivalence pour réunir les deux classes les représentant en une seule.

Voyons comment cela se traduit dans la définition d'un environnement.

Définition 2.2.4. *Un environnement* Γ *est défini par son* support *et sa fonction d'*instanciation *:*

- *le support est la donnée, pour tout* f*, d'une relation d'équivalence sur une partie de* $\mathcal{E}_f$ *; les classes d'équivalence seront appelées des* variables d'environnement *et si* x *est un nom de variable, la variable associée sera notée* $< x >$

- *la fonction d'instanciation est une fonction partielle qui, pour tout f de $\mathcal{F}$, associe à toute variable $< x >$ du support au plus un élément de $\mathcal{D}_f$ noté $\Gamma.< x >$ où f est le nom de trait tel que $x \in \mathcal{E}_f$.*

Nous désignerons les environnements par des majuscules grecques : Γ, Δ,... Comme dans la pratique, le domaine de définition d'un environnement sera toujours fini, nous pourrons aussi définir un environnement par son extension $\{< x_1 >: v_1, \ldots, < x_n >: v_n\}$.

On va définir deux opérations sur les environnements qui seront utiles par la suite.

Définition 2.2.5. *Soit Γ un environnement et $x, y \in \mathcal{E}_f$ deux noms de variable associés au même nom de trait. L'*identification de x et de y dans Γ *produit comme résultat l'environnement $\Gamma[x \equiv y]$ tel que :*

- *son support est obtenu en ajoutant à celui de Γ une équivalence entre x et y ;*
- *sa fonction est alors la même que celle de Γ sauf que :*

$$\Gamma[x \equiv y].< x >= \Gamma[x \equiv y].< y >= \Gamma.< x > \cap \Gamma.< y >$$

Soit Γ un environnement et $x \in \mathcal{E}_f$ une variable. La mise à jour de x dans Γ avec la valeur $v \in \mathcal{D}_f$ *produit l'environnement $\Gamma[x := v]$ de même support, avec la même relation d'équivalence ; sa fonction d'instanciation est la même sauf que $\Gamma[x := v].< x >= v$*

Maintenant, nous allons définir la notion de trait polarisé et de structure de traits polarisés sur un environnement. Par opposition, ce que nous avons défini jusqu'à maintenant sera appelé *trait effectif* et *structure de traits effective*.

Définition 2.2.6. *Un trait polarisé sur un environnement Γ est un triplet $(f, p, < x >)$ où f est un nom de trait, p une polarité et $< x >$ une variable du support de Γ telle que $x \in \mathcal{E}_f$. Le* trait polarisé effectif *correspondant est le trait $(f, p, \Gamma.< x >)$*

Une structure de traits polarisés sur un environnement Γ *est une fonction partielle S qui à tout nom de trait f de $\mathcal{F}$ associe au plus un trait polarisé $(f, p, < x >)$ sur Γ qui sera désigné par $S.f$. La* structure de traits effective *correspondante est obtenue en remplaçant dans S chaque trait polarisé $(f, p, < x >)$ par $(f, p, \Gamma.< x >)$.*

Quand, dans le processus de composition syntaxique, nous allons fusionner des nœuds syntaxiques, nous allons être amenés à unifier les structures de traits polarisés attachées à ces nœuds. Cette opération est classique en l'absence de polarités mais dans le cas de traits polarisés, nous avons besoin de dire comment composer les polarités. Nous allons les additionner selon la table 2.2. Dans cette table, les cases vides signifient que l'addition n'est pas définie pour les polarités correspondantes.

.

Définition 2.2.7. *Soit deux structures de traits S et T. Deux cas sont envisagés :*

- *S et T sont définies sur le même environnement Γ. Supposons :*

$$S = \{(f_1, p_1, < x_1 >), \ldots, (f_n, p_n, < x_n >)\} \cup S_1$$

	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftarrow$		$\leftrightarrow$	$\leftarrow$	
$\rightarrow$	$\leftrightarrow$		$\rightarrow$	
$=$	$\leftarrow$	$\rightarrow$	$=$	$\leftrightarrow$
$\leftrightarrow$			$\leftrightarrow$	

TAB. 2.2 – Table d'addition des polarités

$$T = \{(f_1, q_1, < y_1 >), \ldots, (f_n, q_n, < y_n >)\} \cup T_1$$

où S_1 et T_1 ont des ensembles de définition disjoints.

Le résultat de l'unification de S et T, noté $S + T$, et l'environnement résultant Γ' sont définis par :

$$S + T = \{(f_1, p_1 + q_1, < x_1 >), \ldots, (f_n, p_n + q_n, < x_n >)\} \cup S_1 \cup T_1$$
$$\Gamma' = \Gamma[x_1 \equiv y_1] \ldots [x_n \equiv y_n]$$

– *S est définie sur un environnement Γ et T est une structure de traits effective. Supposons :*

$$S = \{(f_1, p_1, < x_1 >), \ldots, (f_n, p_n, < x_n >)\} \cup S_1$$
$$T = \{(f_1, q_1, v_1), \ldots, (f_n, q_n, v_n)\} \cup T_1$$

où S_1 et T_1 ont des ensembles de définition disjoints.

Le résultat de l'unification de S et T, noté $S + T$, et l'environnement résultant Γ' sont définis par :

$$S + T = \{(f_1, p_1 + q_1, < x_1 >), \ldots, (f_n, p_n + q_n, < x_n >)\} \cup S_1 \cup T_1$$
$$\Gamma' = \Gamma[x_1 := v_1 \cap \Gamma. < x_1 >] \ldots [x_n := v_n \cap \Gamma. < x_n >]$$

Si une des polarités calculées n'est pas définie ou si une des valeurs de trait calculées est égale à $\emptyset$, $S+T$ n'est pas défini et nous disons que S et T ne sont pas unifiables.

Illustrons cette définition en considérant l'environnement Γ suivant :

variable	$< x_1 >$	$< x_2 >$	$< x_3 >$	$< x_4 >$	$< x_5 >$	$< x_6 >$	$< x_7 >$
valeur	*np*	?	*subj*	"*Jean*"	*sg*	*m*	3

Dans cet environnement, l'équivalence entre variables se réduit à l'identité. Soit maintenant les deux structures de traits suivantes définies sur Γ

$$S = \begin{bmatrix} cat \rightarrow < x_1 > \\ funct \leftarrow < x_2 > \\ phon = < x_4 > \\ num = < x_5 > \\ gen = < x_6 > \end{bmatrix} \qquad T = \begin{bmatrix} cat \leftarrow < x_1 > \\ funct \rightarrow < x_3 > \\ num = < x_5 > \\ pers = < x_7 > \end{bmatrix}$$

Pour unifier S et T, nous devons étendre la relation d'équivalence sur Γ en ajoutant $x_2 \equiv x_3$. Nous obtenons l'environnement Γ' suivant :

variable	$< x_1 >$	$< x_2 > \cup < x_3 >$	$< x_4 >$	$< x_5 >$	$< x_6 >$	$< x_7 >$
valeur	*np*	*subj*	"*Jean*"	*sg*	*m*	3

La structure de traits $S + T$ qui est définie sur Γ' se présente ainsi :

$$S + T = \left[\begin{array}{l} cat \leftrightarrow < x_1 > \\ funct \leftrightarrow < x_2 > \cup < x_3 > \\ phon = < x_4 > \\ num = < x_5 > \\ gen = < x_6 > \\ pers = < x_7 > \end{array}\right]$$

Dans la suite de l'exposé, nous ne représenterons plus de façon explicite les environnements, sauf lorsque cela sera nécessaire ; nous représenterons les structures de traits par leurs valeurs effectives avec, lorsque cette valeur est partagée, la variable d'environnement $< x >$ correspondante placée devant la valeur effective.

Les descriptions d'arbres

Comme nous l'avons déjà dit, les objets syntaxiques que nous manipulons sont non pas directement des arbres mais des descriptions d'arbres que nous allons maintenant définir formellement.

Nous désignerons les nœuds syntaxiques des descriptions par des noms. De la même façon que nous aurons besoin dans le processus d'analyse syntaxique d'identifier des variables d'environnement, nous aurons besoin d'identifier des nœuds syntaxiques. Un ensemble de nœuds sera donc défini comme un ensemble de classes d'équivalence de noms de nœuds. Par un abus de langage, nous confondrons en général un nœud avec un de ses noms.

Définition 2.2.8. *Une* description syntaxique D *est la donnée de :*
- *un ensemble* $|D|$ *de nœuds syntaxiques ;*
- *un environnement* Γ *;*
- *une fonction* $Feat_D$ *qui à tout nœud syntaxique* N *de* $|D|$ *associe une structure de traits* $Feat_D(N)$ *sur* Γ *;*
- *un ensemble de relations sur* $|D|$ *de la forme :* $N_1 \prec N_2$, $N_1 > N_2$, $N > \{N_1, \ldots, N_p\}$, $N_1 \overset{\star}{>}_S N_2$ *où* S *est une structure de traits neutres.*

Pour tout nœud N d'une description D où le trait f a une valeur polarisée, nous noterons $p_f(N)$ la polarité du trait et $v_f(N)$ sa valeur effective brute (sans la polarité).

Les relations entre nœuds syntaxiques $\prec$, $>$, $\overset{\star}{>}$ sont des relations de *préséance*, de *parenté* ou de *domination*. On indique que l'ensemble $\{N_1, \ldots, N_p\}$ des fils d'un nœud N donné est clos par la relation $N > \{N_1, \ldots, N_p\}$. Le nombre p de fils peut être réduit à 0 et, dans ce cas, la relation s'écrit : $N > \{\ \}$. Une relation $N_1 \overset{\star}{>}_S N_2$ signifie que N_1 domine N_2 au sens large sous la contrainte S dont la sémantique sera donnée plus loin. Lorsque $S = \{\ \}$, on note alors simplement :$N_1 \overset{\star}{>} N_2$.

Dans la définition d'une description, nous nous sommes cantonnés à certains types de relations mais nous pouvons très bien en imaginer d'autres telles que les relations de préséance immédiate ou de domination stricte.

Par ailleurs, nous nous donnerons une description syntaxique particulière que nous appellerons la *description incohérente* et que nous noterons $\perp$.

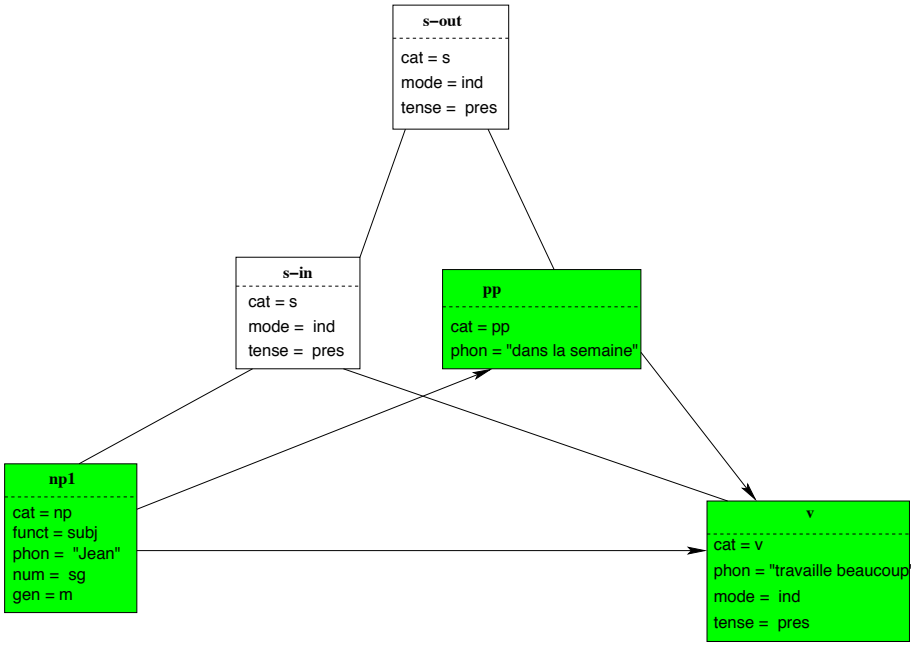


FIG. 2.12 – Arbre syntaxique de *« Jean dans la semaine travaille beaucoup »*

Nous aurions pu aussi présenter les descriptions à l'aide de la logique classique sous forme de formules comme le font J. Rogers et K. Vijay-Shanker [RVS92]. Cela aurait été utile si nous avions eu à raisonner sur les descriptions d'arbres mais ce n'est pas le cas ici.

Une description syntaxique est une façon économe et souple de représenter un ensemble d'arbres syntaxiques par un ensemble de propriétés les caractérisant. Chacun de ces arbres représente alors un modèle de cette description dans un sens semblable à celui que donne la théorie des modèles. Ces modèles possèdent néanmoins des particularités que nous allons maintenant nous attacher à définir.

Tout d'abord, les modèles que nous considérons ici sont des *arbres finis ordonnés*. Un arbre fini a une profondeur finie mais aussi chaque nœud a un nombre fini de fils. Un arbre ordonné est un arbre où les fils de chaque nœud sont partiellement ordonnés et où cet ordre se transmet par héritage aux descendants des fils. Une conséquence linguistique de cette restriction des modèles à des arbres ordonnés est que les syntagmes correspondront nécessairement à des segments continus de la phrase. Par exemple, il n'est pas possible d'analyser la phrase *« Jean dans la semaine travaille beaucoup »* selon l'arbre donné par la figure 2.12[8]. Cet arbre montre que *« Jean ... travaille beaucoup »* constitue un syntagme discontinu représenté par le nœud *s-in* et que la discontinuité est liée au fait que l'ordre de préséance entre les nœuds ne fait pas de l'arbre un arbre ordonné où toute relation d'ordre devrait provenir par héritage d'une relation d'ordre entre deux frères. Le fait de se restreindre à des modèles qui

[8]Nous avons simplifié l'arbre en considérant les syntagmes *« travaille beaucoup »* et *« dans la semaine »* comme des items lexicaux.

soient des arbres finis ordonnés nous limite donc quant au pouvoir d'expression du formalisme.

En plus d'être finis et ordonnés, les arbres que nous considérons comme modèles sont étiquetés : chaque nœud est étiqueté par une structure de traits de la forme $f = v$ où f est un élément de $\mathcal{F}$ et où v est un élément de $\mathcal{V}_f$. Un arbre fini ordonné étiqueté de cette façon sera appelé un *arbre syntaxique*. Un arbre syntaxique est équipé des relations de préséance, de parenté, de domination et d'égalité définis canoniquement par sa structure.

Nous sommes maintenant en mesure de donner une définition précise d'un modèle d'une description syntaxique.

Définition 2.2.9. *Étant donné une description syntaxique D, un modèle de cette description est un couple (T, I) où T est un arbre syntaxique et I une fonction d'interprétation de $|D|$ dans l'ensemble $|T|$ des nœuds de T qui vérifie les conditions suivantes :*

1. *I préserve toutes les relations de préséance, de parenté et de domination de D ;*
2. *pour tout nœud N de $|D|$ et pour tout nom de trait f présent dans la structure de traits $Feat_D(N)$ associée à N, la structure de traits associée à $I(N)$ contient un trait $f = v$ tel que $v \leq v_f(N)$;*
3. *pour toute variable $< x >$ de l'environnement Γ associé à D appartenant à $\mathcal{E}_f$, si les structures de traits associées à deux nœuds quelconques N et M contiennent respectivement les traits $(f, p, < x >)$ et $(f, q, < x >)$ alors $I(M)$ et $I(N)$ sont étiquetés par une structure de traits contenant le même trait de la forme $f = v$.*
4. *si D contient un prédicat $N > \{N_1, \ldots, N_p\}$, les fils de $I(N)$ sont les nœuds $I(N_1), \ldots, I(N_p)$ tous distincts.*
5. *si D contient un prédicat $N_1 \overset{\star}{>}_S N_2$, alors tout nœud de T qui domine strictement $I(N_2)$ et qui est dominé par $I(N_1)$ au sens large doit être étiqueté par une structure de traits unifiable avec S.*

Bien entendu, la description incohérente n'a pas de modèle.

La notion de modèle telle qu'elle vient d'être définie ignore les polarités. Il est donc nécessaire de la compléter pour donner un sens à cette notion et exprimer le fait que chaque trait positif doit rencontrer un trait opposé pour être neutralisé. C'est ce que réalise exactement la notion de *modèle neutre*.

Définition 2.2.10. *Un modèle (T, I) d'une description D est* neutre *si pour tout nœud N de $|T|$ et pour tout trait f de $\mathcal{F}$, l'ensemble $\mathcal{N}_f$ des nœuds de $|D|$ interprétés par N où f a une valeur a la propriété suivante : $\Sigma_{N' \in \mathcal{N}_f} p_f(N') \in \{=, \leftrightarrow\}$.*

Maintenant, il est facile de comprendre que toute description a une infinité de modèles et qu'il est nécessaire de privilégier certains d'entre eux que nous appellerons des *modèles minimaux*. Il s'agit de modèles qui n'ajoutent aucune information supplémentaire par rapport à celle qui est incluse dans la description correspondante.

Définition 2.2.11. *Un modèle (T, I) d'une description D est* minimal *si tout nœud de $|T|$ interprète un nœud de $|D|$, si toute relation de parenté de T interprète une relation de parenté de D et si pour tout trait $f = v$ associé à un nœud N de $|T|$, il existe un nœud M de $|D|$ tel que $I(M) = N$ et tel que $Feat_D(M)$ contienne un trait f.*

Par exemple, la description syntaxique de la figure 2.13 a un unique modèle neutre minimal qui est l'arbre syntaxique présenté sur la figure 2.14. La fonction d'interprétation est représentée en mettant dans l'entête des nœuds du modèle les noms des nœuds de la description qu'ils interprètent séparés par un point.

Raffinement et équivalence de descriptions syntaxiques

La notion de modèle neutre et minimal d'une description permet de définir une relation de raffinement et une relation d'équivalence entre descriptions syntaxiques.

Définition 2.2.12. *Une description D_1 est un* raffinement *d'une description D_2 si tout arbre syntaxique qui est un modèle neutre et minimal de D_1 est un modèle neutre et minimal de D_2. D_1 et D_2 sont* équivalentes *si elles sont des raffinements l'une de l'autre.*

Choisissons comme description D_1 celle qui est présentée sur la figure 2.13 et choisissons comme description D_2 la description D_1 d'où nous avons retiré les relations de préséance entre les ancres. Cela revient à considérer les mots de la phrase sans fixer leur ordre. Il est facile de montrer que D_1 est équivalente à D_2 du fait qu'il y a une seule façon d'ordonner les mots pour former une phrase correcte.

Maintenant que nous avons défini la notion de descriptions syntaxiques équivalentes, nous allons pouvoir parler de simplification d'une description : simplifier une description c'est la remplacer par une description équivalente plus simple.

Ce qui est intéressant c'est d'exhiber des règles de simplification faciles à mettre en œuvre. Pour cela, il est nécessaire de définir préalablement l'opération de *fusion de deux nœuds* d'une description donnée qui va intervenir dans certaines simplifications. Elle servira aussi à réaliser l'opération de neutralisation de traits.

Définition 2.2.13. *Soit deux nœuds N et M d'une description syntaxique D Si $Feat_D(N)$ et $Feat_D(M)$ ne sont pas unifiables, le résultat de la fusion de N et M est la description incohérente $\perp$.*

Sinon le résultat de la fusion de N et M est une description obtenue de la façon suivante : on remplace dans l'ensemble des nœuds présents dans D les nœuds N et M par $N \cup M$[9] et on associe comme structure de traits à ce nouveau nœud $Feat_D(N) + Feat_D(M)$.

[9]Si nous considérons que N et M représentent des nœuds, c'est-à-dire des ensembles de noms de nœuds, $N \cup M$ représente l'union de leurs noms qui réalise l'identification des deux nœuds.

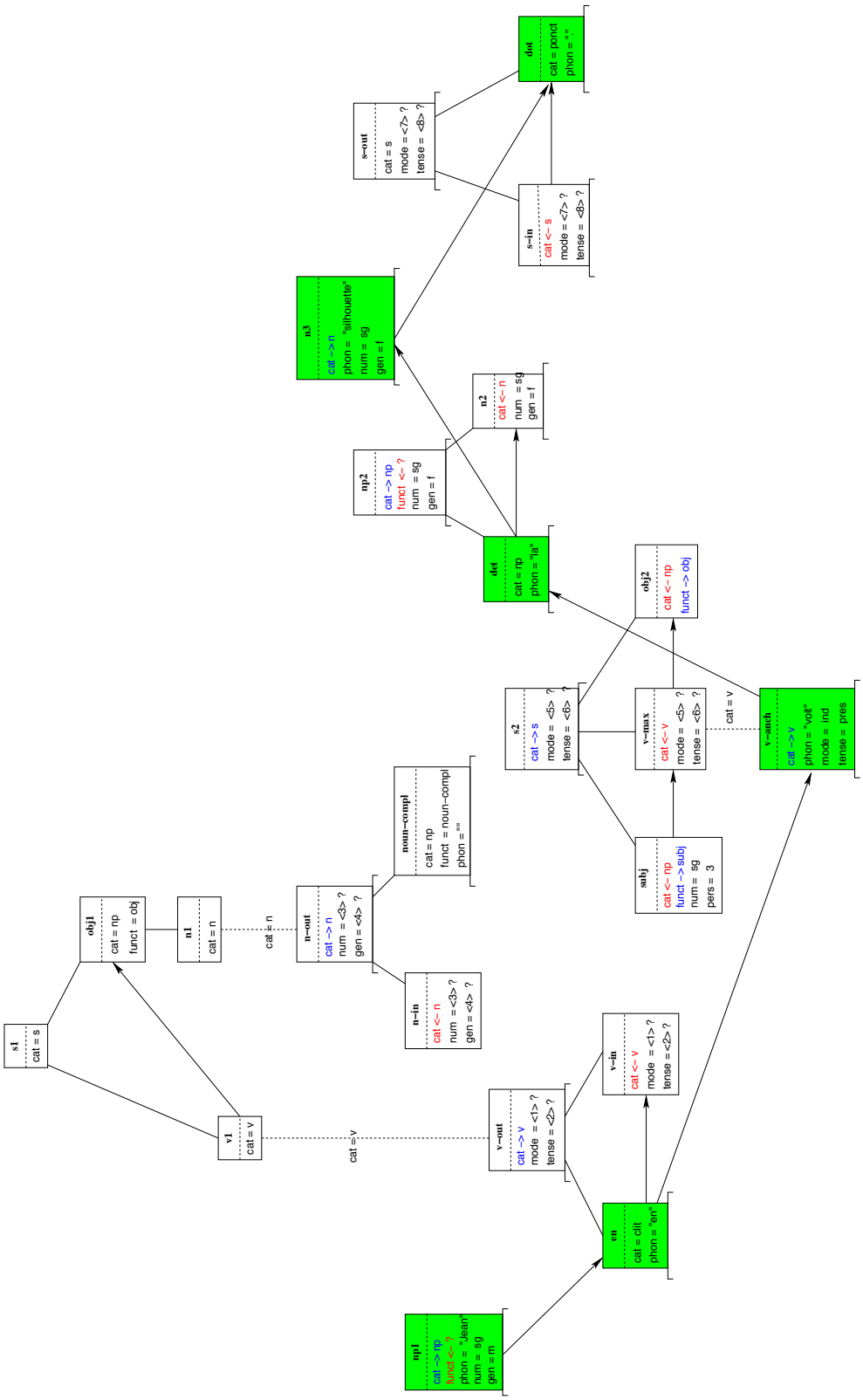


FIG. 2.13 – Description exprimant la spécification syntaxique de la phrase *« Jean en voit la silhouette. »*

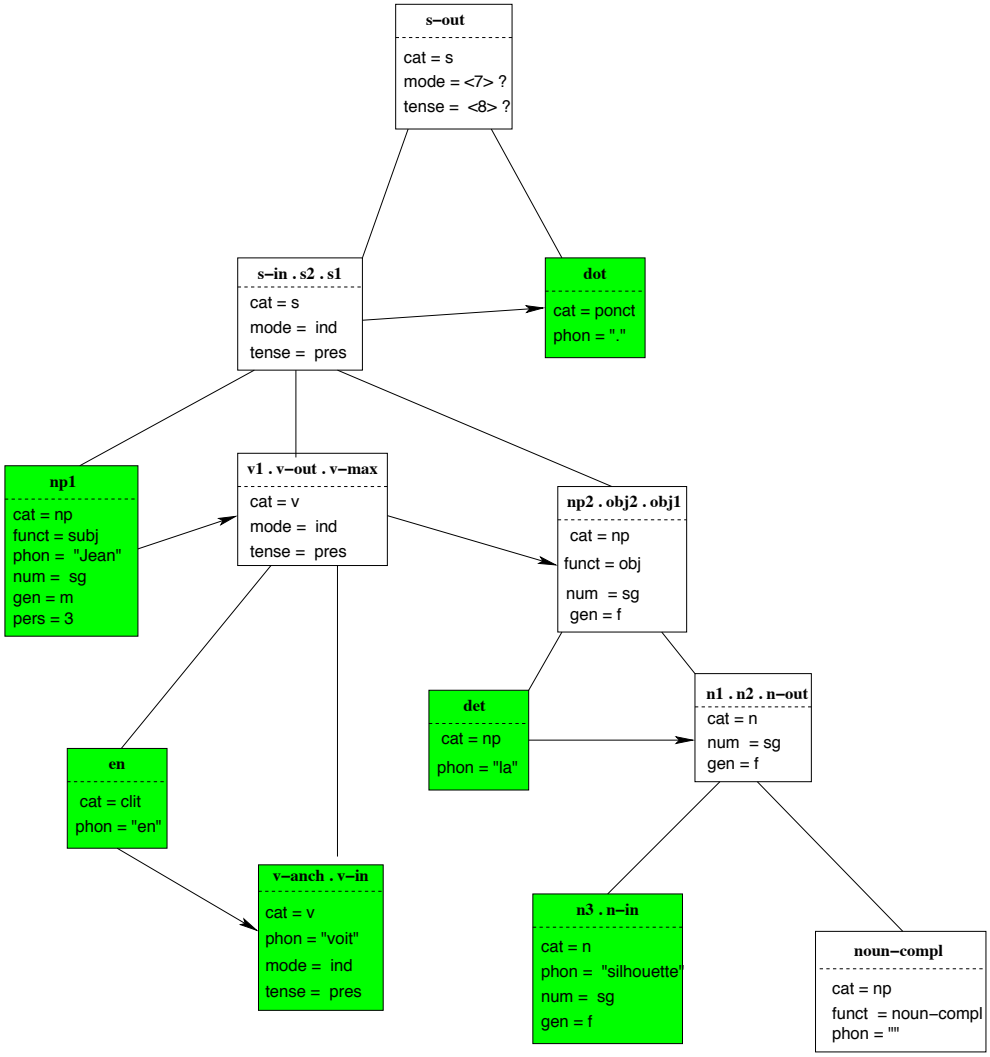
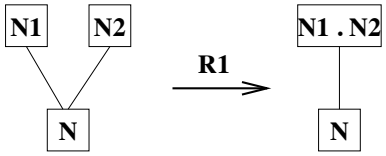


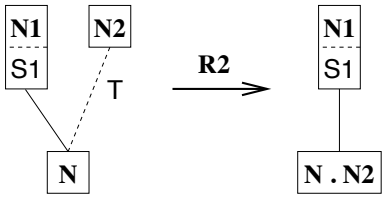
FIG. 2.14 – Arbre syntaxique de la phrase *« Jean en voit la silhouette. »*

Présentons maintenant quelques règles de simplification. Nous choisirons celles qui nous serons le plus utiles par la suite tout en sachant que nous pourrions en mettre en évidence bien d'autres. Nous laisserons le soin au lecteur, s'il le souhaite, de démontrer qu'il s'agit vraiment de règles de simplification, c'est-à-dire qu'elles transforment une description en une description équivalente. Pour indiquer que nous sommes en présence d'une relation de la forme $N_1 > N_2$ ou $N_1 > \{N_2, \ldots\}$, nous utiliserons l'abréviation : $N_1 \rhd N_2$.

R1 unicité du père : si une description contient deux relations de la forme $N_1 \rhd N$ et $N_2 \rhd N$, on fusionne N_1 et N_2.

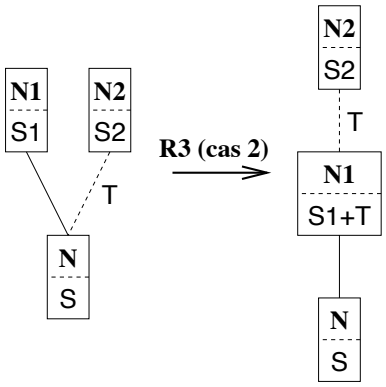


R2 transformation d'une domination en égalité : si une description contient deux relations de la forme $N_1 \rhd N$ et $N_2 \overset{\star}{>}_T N$ et si la structure de traits S_1 associée à N_1 par D n'est pas unifiable avec T, alors on peut supprimer la relation $N_2 \overset{\star}{>}_T N$ et fusionner N_2 et N.



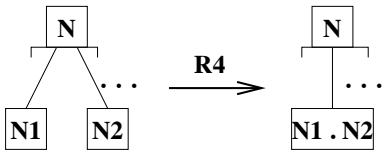
R3 transformation d'une domination en domination stricte : si une description contient deux relations de la forme $N_1 \rhd N$ et $N_2 \overset{\star}{>}_T N$ et si les structures de traits S et S_2 associées par D à N et N_2 ne sont pas unifiables, alors deux cas sont possibles :

1. si S_1 (associé à N_1) et T ne sont pas unifiables, on remplace D par la description incohérente $\perp$;
2. sinon on remplace dans Γ par l'environnement Γ' obtenu en unifiant S_1 et T, la structure de traits associée à N_1 par $S_1 + T$ et la relation $N_2 \overset{\star}{>}_T N$ par $N_2 \overset{\star}{>}_T N_1$.



R4 égalité forcée entre fils : si une description contient deux relations, l'une de la forme $N \rhd N_1$ et l'autre de la forme $N > \{N_2, \ldots\}$ et si

N_2 est le seul nœud de l'ensemble $\{N_2, \ldots\}$ dont la structure de traits est unifiable avec celle de N_1, alors on fusionne N_1 et N_2.



2.2.2 La neutralisation de traits opposés comme principe de l'analyse syntaxique

L'opération de neutralisation de traits opposés

L'opération de *neutralisation de traits opposés* est l'opération fondamentale qui va permettre de réaliser la composition syntaxique des mots d'une phrase et par là même l'analyse syntaxique de cette phrase.

Cette opération peut être vue comme une fusion particulière de deux nœuds dans le cas où ceux-ci sont porteurs de traits opposés. En toute rigueur, elle peut être définie ainsi :

Définition 2.2.14. *Si une description D associe à deux nœuds N et M deux structures de traits contenant respectivement des traits polarisés opposés de la forme $f \rightarrow < x >$ et $f \leftarrow < y >$, la neutralisation de ces traits opposés consiste à fusionner les nœuds N et M.*

Si D' est la description qui résulte de cette fusion, nous noterons ceci de la façon suivante :$D \overset{N \overset{f}{=} M}{\longrightarrow} D'$.

La correction de cette opération est établie par la proposition suivante :

Proposition 2.2.1. *Si D_2 est une description qui est le résultat d'une neutralisation sur une description D_1, alors D_2 est un raffinement de D_1.*

Remarque 2.2.1. *Une neutralisation sur une description D_1 peut très bien déboucher sur une description D_2 incohérente si les deux nœuds qui doivent fusionner ont des traits de même nom et de même polarité non neutre ou de même nom et de valeurs non unifiables.*

La proposition ci-dessus éclaire le sens de l'opération de neutralisation. Itérée, cette dernière va permettre de spécifier progressivement une description et on est sûr que tous les modèles neutres et minimaux de la description finale à laquelle on aboutira seront des modèles neutres et minimaux de la description initiale.

Maintenant, la neutralisation de traits opposés n'aurait guère d'intérêt sans une propriété de complétude que nous allons maintenant énoncer.

Proposition 2.2.2. *Si une description D_1 contient au moins un trait positif ou négatif (nous dirons qu'elle n'est pas neutralisée), alors pour tout modèle neutre et minimal (T, I) de D_1, il existe une neutralisation sur D_1 qui aboutisse à une description D_2 qui ait comme modèle neutre et minimal (T, I).*

Les propriétés de complétude et de correction de l'opération de neutralisation de traits opposés vont permettre d'envisager l'analyse syntaxique sous un angle opérationnel et non plus seulement sous une angle déclaratif.

Le processus d'analyse syntaxique

Les grammaires d'interaction sont lexicalisées. Une grammaire particulière est définie par un lexique qui associe à chaque mot de la langue un ensemble de descriptions syntaxiques. Dans chacune de ces descriptions D, un nœud particulier est distingué comme *ancre* de D et il est noté $Anchor(D)$. Il représentera le syntagme constitué par le mot auquel est associée la description D.

Faire l'analyse syntaxique d'une phrase $w_1 \dots w_n$ à l'aide d'une grammaire d'interaction, c'est d'abord sélectionner pour chacun des mots w_i une description syntaxique D_i extraite du lexique. La suite $D_1, \dots, D_n$ constituera ce que nous appellerons un *étiquetage syntaxique* de la phrase $w_1 \dots w_n$.

Nous supposons que les descriptions $D_1, \dots, D_n$ ont leurs ensembles de noms de nœuds qui sont disjoints (sinon nous effectuons des renommages). Nous faisons la même hypothèse pour les noms de leurs variables d'environnement. Alors, il est facile de définir la description $D_1 \cup \dots \cup D_n$. Son support est $|D_1| \cup \dots \cup |D_n|$ et son environnement est obtenu comme réunion des environnements des descriptions $D_1, \dots, D_n$. A chacun de ses nœuds, on associe la structure de traits qu'il a dans la description D_i à laquelle il appartient et, pour ce qui est des relations entre nœuds, on effectue l'union des relations présentes dans les D_i.

Ensuite, il faut compléter la description $D_1 \cup \dots \cup D_n$ par des relations de préséance entre les ancres des D_i pour exprimer l'ordre des mots dans la phrase. On ajoute donc les relations $Anchor(D_1) \prec Anchor(D_2)$, $Anchor(D_{n-1}) \prec Anchor(D_n)$ et on obtient la description D_0 qui va constituer le point de départ de l'analyse syntaxique proprement dite. D_0 représente ce qu'on pourrait appeler une spécification syntaxique de la phrase $w_1 \dots w_n$. La figure 2.13 en annexe montre un exemple de spécification syntaxique de la phrase *« Jean en voit la silhouette. »*. Les ancres y figurent en grisé.

L'analyse proprement dite consiste à trouver tous les modèles neutres et minimaux de D_0. Jusqu'à présent, nous sommes seulement capables de dire si un arbre syntaxique donné est un modèle neutre et minimal d'une telle description mais ce qui est important c'est de disposer de méthodes qui nous permettent de calculer ces modèles. C'est l'opération de neutralisation de traits opposés qui va servir de base au calcul de modèles. Une analyse syntaxique va se ramener à une suite maximale de neutralisations, suite qui est nécessairement finie, et deux cas sont alors possibles :

- il existe une telle suite qui se termine par une description qui n'est pas neutralisée et alors toutes les autres qui n'aboutissent pas à la description incohérente se terminent aussi par une description qui n'est pas neutralisée ; on peut conclure que la description initiale n'a pas de modèle neutre donc la phrase analysée est rejetée par la grammaire ;
- un certain nombre d'analyses mènent à des descriptions neutralisées non incohérentes ; les modèles minimaux de ces descriptions vont constituer des modèles neutres minimaux de la description initiale.

En théorie, dans le second cas, le calcul de modèles minimaux de descriptions neutres peut être coûteux mais tout l'intérêt est de se trouver dans des situations particulières où ce calcul devient trivial. C'est notamment le cas si l'on réussit à éliminer toute relation de domination d'une description. Théorique-

ment, une neutralisation n'entraîne aucune suppression de relation dans la description où elle se passe mais en transformant la description elle va permettre éventuellement de la simplifier. C'est l'application des règles de simplification qui va entraîner la réalisation de certaines relations de domination, ce qui va permettre de les supprimer.

Reprenons l'exemple de l'analyse syntaxique de la phrase *« Jean en voit la silhouette. »* en partant de la description syntaxique D_0 de la figure 2.13. Nous pouvons choisir de commencer l'analyse en composant *en voit* à partir de *en* et de *voit*.

Nous allons réaliser cette composition syntaxique en commençant par neutraliser les traits $cat \rightarrow v$ et $cat \leftarrow v$ des nœuds respectifs *v-anch* et *v-in*, c'est-à-dire par fusionner les deux nœuds. Dès que nous effectuons une neutralisation, nous essayons de simplifier la description obtenue par l'application de l'une ou plusieurs des 4 règles précédemment exposées. Dans notre cas, la fusion de *v-anch* et *v-in* entraîne une application de la règle $R3$ et transformation de la cible de la relation de domination ayant pour source *v-max* : la cible devient *v-out*.

Maintenant réalisons la neutralisation des traits $cat \rightarrow v$ et $cat \leftarrow v$ de *v-out* avec *v-max*. La fusion de ces deux nœuds permet ensuite d'appliquer la règle de simplification $R2$ qui entraîne la fusion des nœuds *v1* et *v-out.v-max*. Après, nous pouvons enchaîner par l'application de la règle $R1$ qui entraîne la fusion des nœuds *s1* et *s2*. Enfin, l'application de la règle $R4$ permet de fusionner les nœuds *obj1* et *obj2*. Nous obtenons alors une description D_1 qui se présente comme l'indique la figure 2.15.

Ce petit exemple nous montre comment une neutralisation peut entraîner une application en chaîne de règles de simplification. De cette façon, il est possible de faire de la superposition partielle d'arbres, chose que ne font pas la plupart des formalismes linguistiques manipulant des arbres. Nous avons déjà évoqué cet atout des grammaires d'interaction dans la sous-section 2.1.1.

2.3 Exemples de modélisation de phénomènes syntaxiques

Nous avons déjà vu comment les grammaires d'interaction pouvaient rendre compte de phénomènes comme les barrières à l'extraction, la négation ou l'inversion du sujet dans les relatives. Pour continuer à illustrer le pouvoir d'expression des grammaires d'interaction, nous voudrions évoquer quelques phénomènes syntaxiques un peu délicats en français : la dépendance non bornée en cascade dans les propositions relatives[10], l'ordre des pronoms personnels clitiques et le comportement de ces pronoms par rapport aux auxiliaires de temps et aux verbes causatifs et perceptifs.

[10]Ce phénomène correspond à ce qui est qualifié en anglais de *pied-piping* et je remercie Gérard Huet de m'avoir suggéré ce terme français plus parlant.

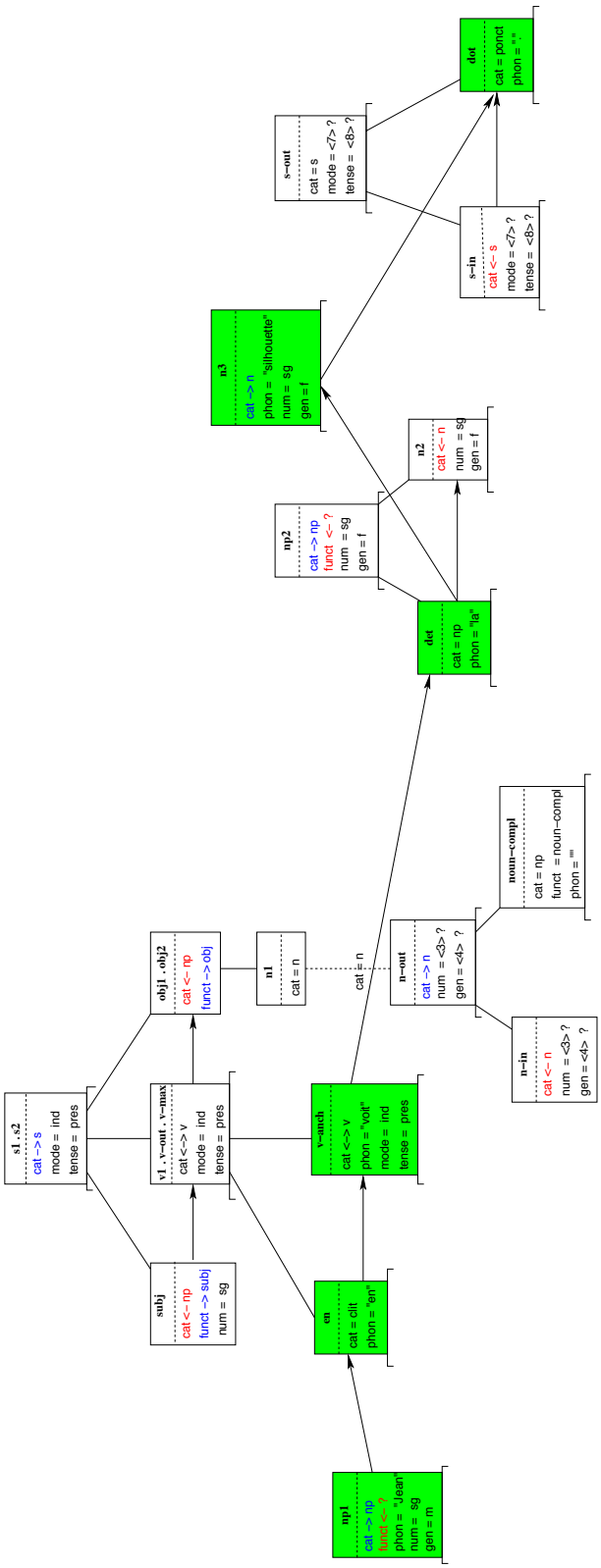


FIG. 2.15 – Raffinement de la description syntaxique de la figure 2.13 par composition de *en* avec *voit*.

2.3.1 La dépendance non bornée en cascade dans les propositions relatives

Ce phénomène peut être décrit de la façon suivante. Une proposition relative introduit une dépendance non bornée entre son antécédent et le verbe qu'elle contient et qui gouverne cet antécédent. Par exemple, dans la phrase *« l'homme que Jean voit dort »*, l'antécédent de la proposition relative *« que Jean voit »* est *« l'homme »* et il dépend du verbe *voit* comme complément d'objet direct. On peut imaginer que cet antécédent a été extrait de sa position canonique d'objet et qu'il y laisse une trace. Cette dépendance est non bornée parce que sa distance peut être augmentée par une imbrication au sein de la relative de complétives successives comme dans l'exemple *« l'homme que Marie dit que Jean pense voir dort »*.

Une seconde dépendance non bornée vient s'enchaîner à la première et prend son origine dans la différenciation entre l'antécédent de la proposition relative et le syntagme extrait. Par exemple, dans l'expression *« l'homme à la mère de qui je parle »* l'antécédent est *« l'homme »* et le syntagme extrait est *« à la mère de qui »*. Ce syntagme se rattache à *parle* dans une dépendance non bornée et il y a une seconde dépendance non bornée entre l'antécédent *l'homme* et le nom *mère* si on considère celui-ci comme la tête de *« à la mère de qui »*.

Ce phénomène existe aussi dans les propositions interrogatives comme l'illustre l'exemple suivant : *« A la mère de quel homme Pierre croit-il que Jean parle ? »* et il se traite de façon analogue.

Dans HPSG, C. Pollard et I. Sag [PS94] propose de représenter les deux dépendances non bornées à l'aide de deux traits non locaux :

- le trait *SLASH* permet de faire le lien entre la trace et le syntagme extrait ;
- le trait *REL* permet de faire le lien entre le syntagme nominal interrogé et l'antécédent.

Ces traits remontent de proche en proche selon le principe des traits non locaux jusqu'à être capturés, le premier par l'entrée lexicale de la tête de la proposition relative et le second par l'entrée lexicale de la tête du syntagme extrait.

En TAG, A. Joshi et A. Kroch [KJ85, Kro87] proposent de représenter la dépendance liant le syntagme extrait au verbe de la relative dont il dépend par un arbre constituant une entrée lexicale particulière de ce verbe et l'extension de la dépendance est obtenue par des adjonctions prédicatives successives. La seconde dépendance non bornée est exprimée à l'aide d'un trait particulier qui remonte du mot relatif au syntagme extrait au fur et à mesure de la composition de ce dernier. S. Kahane, M.-H. Candito et Y. de Kercadiao [KCdK00] ont montré qu'une telle représentation n'est pas cohérente avec les dépendances sémantiques correspondantes et propose une représentation où la construction de la relative est plutôt rattachée au mot relatif qu'à un verbe. Les auteurs admettent toutefois que les TAG ne sont pas forcément le meilleur cadre pour formuler cette représentation qui en montre plutôt les limites.

Nous proposons aussi d'attacher la construction de la proposition relative au mot relatif qui lui est associé. Alors, pour représenter les deux dépendances non bornées d'une manière uniforme, nous utilisons les relations de domination

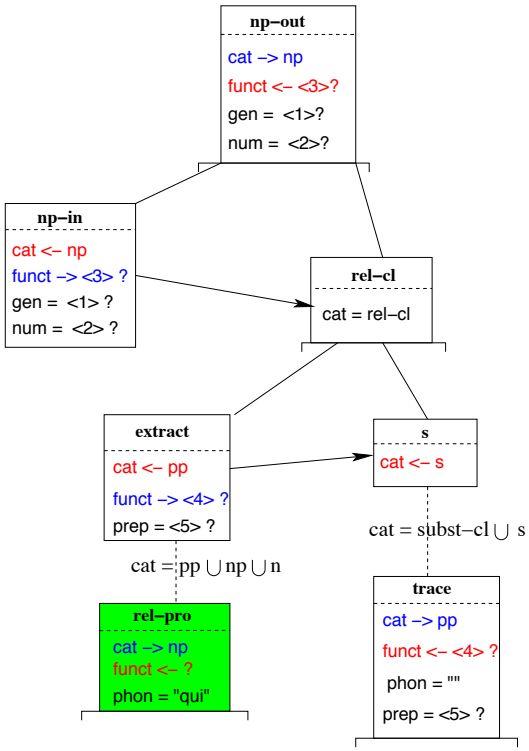


FIG. 2.16 – Entrée lexicale du pronom relatif *qui* prenant en compte la dépendance non bornée en cascade

contraintes. La figure 2.16 montre une entrée lexicale possible pour le pronom relatif *qui* illustrant cette proposition. Nous laissons de côté l'utilisation de *qui* comme sujet de la proposition relative qu'il introduit et nous supposons que la proposition relative est un modifieur du syntagme nominal qui est son antécédent. La dépendance non bornée entre l'antécédent et le syntagme extrait est représentée par la domination du nœud *extract* sur le nœud *rel-pro* et celle entre le syntagme extrait et le verbe dont il dépend dans la proposition relative est représentée par la domination du nœud *s* sur le nœud *trace*. Les contraintes sur la première relation de domination font que le mot interrogatif ne peut être imbriqué que dans une suite de compléments de noms alors que les contraintes sur la seconde modélisent en creux les barrières à l'extraction.

On retrouve une représentation similaire dans LFG où les deux relations de domination sous-spécifiées se traduisent par deux équations d'incertitude fonctionnelle [Fal01].

2.3.2 L'ordre des pronoms personnels clitiques

Pour simplifier l'exposé, nous nous en tiendrons seulement aux pronoms personnels clitiques placés avant le verbe dans les phrases déclaratives. Nous ne prendrons pas en compte les phrases interrogatives et injonctives. Pour régler la position relative de ces pronoms, on considère en général 6 places possibles avant le verbe ordonnées de gauche à droite et chaque pronom clitique occupe

6	5	4	3	2	1
je	*me*				
tu	*te*				
il	*se*	*le*	*lui*	*y*	*en*
elle		*la*			
on					
nous	*nous*				
vous	*vous*				
ils		*les*	*leur*		

TAB. 2.3 – Ordre des pronoms personnels clitiques en français

une place et une seule. Le tableau 2.3 donne les places possibles. La colonne 6 contient les pronoms personnels sujets tandis que les autres contiennent les pronoms personnels compléments. Chaque place ne pouvant être occupée que par un pronom, deux clitiques figurant dans la même colonne ne peuvent pas être présents en même temps. En outre, les positions 5 et 3 ne peuvent pas être occupées en même temps.

Une première façon de modéliser ces pronoms personnels clitiques est de les considérer comme des affixes du verbe. C'est ce que fait P. Miller et I. Sag avec HPSG [MS97] et qui a été repris pour les TAG par M.-H. Candito [Can99]. Dans le lexique, à côté des formes pleines des verbes, nous trouvons leur forme cliticisée. Par exemple, à côté de *présenterai*, nous aurons *je présenterai*, *me présenterai*, *te présenterai*, *je me présenterai* etc... Des règles lexicales permettent d'obtenir ces formes cliticisées à partir de la forme canonique et ce sont elles qui régissent en même temps l'ordre des clitiques. A. Abeillé [Abe02] propose une représentation très voisine avec des entrées particulières pour les verbes cliticisés où les positions des clitiques sont indiquées par des nœuds de substitution.

Une alternative consiste à ne pas considérer d'entrées spéciales pour les verbes cliticisés mais à attacher dans le lexique les descriptions syntaxiques modélisant le comportement des clitiques à ces derniers considérés comme entités syntaxiques indépendantes. A. Abeillé a fait une proposition de ce genre pour les TAG [Abe91] où les clitiques sont associés à des arbres auxiliaires s'adjoignant sur les verbes à cliticiser. Le principal défaut de cette proposition est qu'en cas de montée elle ne permet pas d'établir simplement le lien entre le clitique et son gouverneur, le verbe qui l'a requis comme complément.

Nous nous proposons aussi de montrer qu'il est possible de représenter les clitiques comme des entités syntaxiques indépendantes qui ont leurs propres entrées lexicales. Nous ne rencontrerons pas les mêmes difficultés qu'a eu A. Abeillé avec les TAG car la souplesse des grammaires d'interaction permet d'exprimer la dépendance syntaxique entre un clitique et son gouverneur même lorsqu'elle n'est pas locale.

Pour modéliser l'ordre relatif des clitiques, nous allons raffiner la catégorie grammaticale v en 7 sous-catégories : v_0, v_1, v_2, v_3, v_4, v_5, v_6. La catégorie v_0 représentera un verbe nu, v_1 un verbe nu précédé d'un clitique en position 1, v_2 un verbe nu précédé d'un clitique en position 2 et éventuellement d'un

autre en position 1 et ainsi de suite.

Voyons maintenant comment représenter les descriptions syntaxiques associées aux verbes et au clitiques. Supposons que nous voulions analyser la phrase *« ils lui en parlent. »*. La figure 2.17 nous montre la description syntaxique initiale qui pourrait être fournie par un lexique pour une telle phrase. La valeur v pour *cat* est une abréviation pour $v_0 \cup v_1 \cup v_2 \cup v_3 \cup v_4 \cup v_5 \cup v_6$. Les descriptions syntaxiques associées aux 3 pronoms clitiques reproduisent toutes le même schéma. Soit un clitique qui est dans la colonne i du tableau 2.17. Le nœud $v - in_i$ représente le verbe auquel il va s'adjoindre. La valeur du trait *cat* est telle que ce verbe ne peut porter que des clitiques qui sont dans des colonnes $j < i$ du tableau 2.17. Le verbe, après adjonction du clitique est représenté par le nœud $v - out_i$. La valeur de son trait *cat* est v_i, ce qui fait que les seuls clitiques qui pourront s'adjoindre ultérieurement sont ceux des colonnes $j > i$ du tableau 2.17.

Pour exprimer l'incompatibilité des positions 5 et 3, le nœud $v - in_5$ porte un trait $cat \leftarrow v_0 \cup v_1 \cup v_2 \cup v_4$ au lieu de $cat \leftarrow v_0 \cup v_1 \cup v_2 \cup v_3 \cup v_4$. En principe, cela n'exclut pas d'avoir conjointement un clitique en position 5 et un clitique en position 3, à condition d'en avoir un troisième entre les deux en position 4 mais une telle éventualité est exclue, compte tenu de la sous-catégorisation des verbes en français. Nous obtenons ainsi le comportement des clitiques que nous souhaitions pour ce qui est de la façon dont ils s'ordonnent.

Maintenant, il est nécessaire de conserver la dépendance syntaxique entre un clitique et le verbe qui le contrôle, notamment dans une perspective d'en extraire une dépendance sémantique. Or, on remarque que dans la description syntaxique associée à chaque clitique, une trace vide joue le rôle du complément en position canonique qui a été cliticisé. C'est donc via cette trace que s'établit la dépendance syntaxique.

2.3.3 La « montée » des clitiques

Quelle que soit la représentation choisie, le lexique associe un clitique au verbe qui le requiert comme complément et que l'on peut appeler son gouverneur, mais ensuite, dans un énoncé, les deux peuvent être séparées par un phénomène de « montée des clitiques ». Le verbe qui reçoit le clitique, le verbe hôte, peut se distinguer du verbe gouverneur et se placer plus haut dans l'arbre syntaxique. M.-H. Candito [Can99] cite à ce sujet la phrase *« il semble l'avoir fait »* où *fait* est le gouverneur de *il* et de *l'* mais ceux-ci ont comme hôtes respectifs *semble* et *avoir*. Toute la difficulté est de prendre en compte ces mouvements. Voici quelques exemples qui donnent une idée du problème :

(a) ils nous en ont parlé
(b) Jean le lui fait manger
(c) Jean le fait se laver
(d) Jean nous entend lui chanter une chanson

Dans la phrase (a), c'est l'auxiliaire de temps *ont* qui a provoqué la montée des clitiques *nous* et *en* que gouverne *parlé*.

Dans la phrase (b), c'est le verbe causatif *fait* qui a provoqué la montée du clitique *le*. Pour le pronom *lui*, on ne peut pas parler de montée car il est gouverné par *fait*.

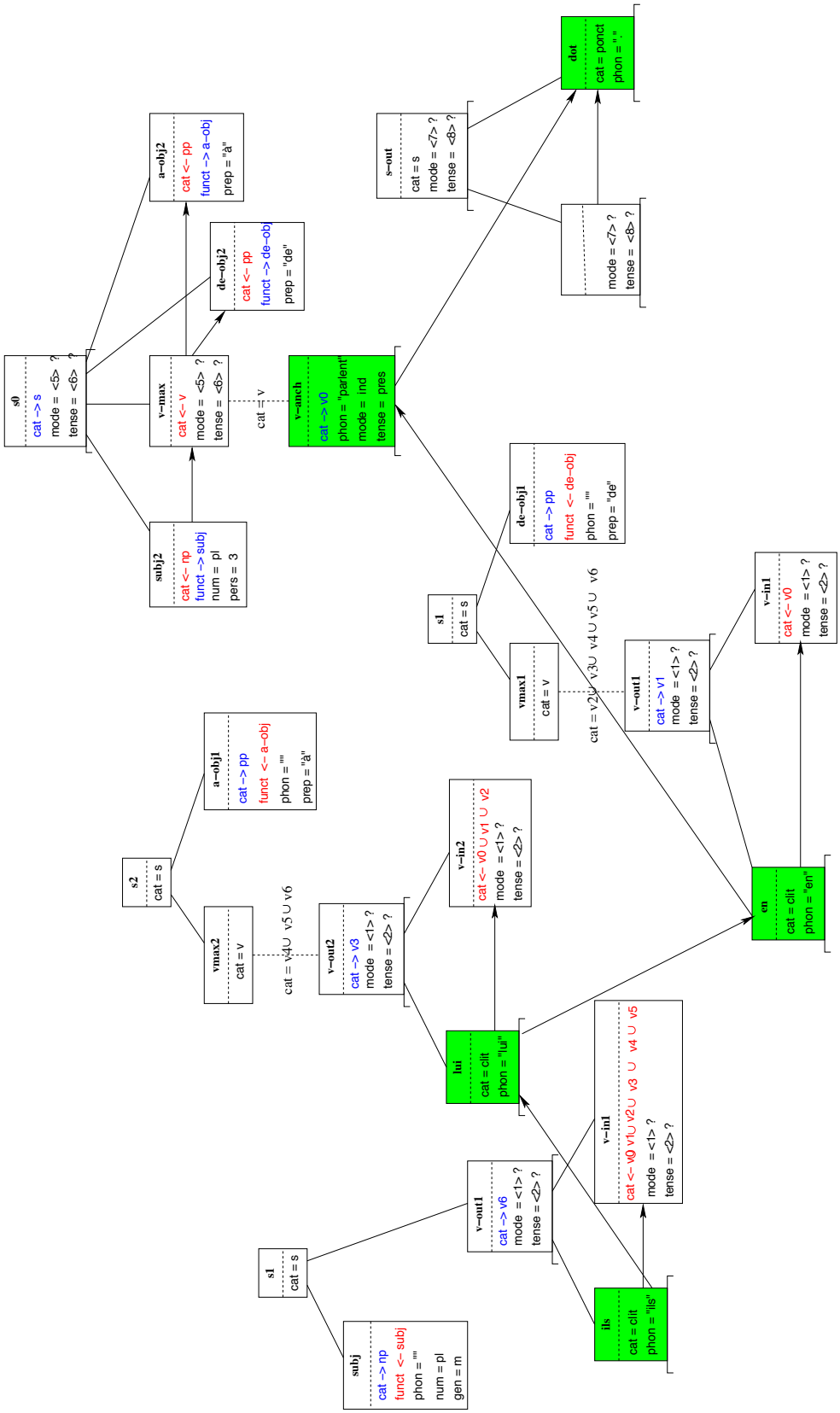


FIG. 2.17 – Description syntaxique fournie par un lexique pour la phrase *« ils lui en parlent. »*

La phrase (c) met en évidence une exception à la montée des clitiques objets provoquée par les causatifs : ce sont les pronoms réfléchis.

Enfin, dans la phrase (d), en présence du verbe perceptif *entend*, on peut discuter de la montée ou pas de *nous*, selon qu'on le considère comme objet de *entend* ou sujet de *chanter*.

On observe aussi ce phénomène de dépendance non locale entre un clitique et son gouverneur lorsque ce dernier n'est pas un verbe mais un adjectif ou un nom prédicatifs comme dans les exemples suivants :

Jean y devient attentif.
Marie en est la sœur.

Nous laisserons de côté ces cas où les gouverneurs ne sont pas des verbes et nous nous limiterons aux cas où ces gouverneurs dépendent d'auxiliaires de temps, de verbes causatifs et perceptifs.

Ces phénomènes ont été étudiés en détail et modélisés dans HPSG par A. Abeillé, D. Godard, P. Miller et I. Sag [AGMS98]. Deux types de représentation en ressortent :

- la composition plate dans laquelle le verbe tête, le verbe complément et les propres compléments de ce dernier se placent à un même niveau pour former un unique syntagme verbal ;
- la composition hiérarchique dans laquelle le verbe complément et ses propres compléments forme un premier syntagme verbal qui vient ensuite compléter le verbe de tête pour former un syntagme verbal de niveau supérieur.

Il en ressort aussi que, pour la composition des auxiliaires de temps avec un participe passé, seul le premier type de composition se justifie alors que, pour les verbes causatifs et les verbes perceptifs, certains arguments plaident en faveur de la structure plate et d'autres en faveur de la structure hiérarchique.

La modélisation dans HPSG de ces représentations linguistiques, telle qu'elle est présentée dans [AGMS98], s'appuie sur une règle lexicale (PRAF-LR) qui permet de cliticiser tout complément d'un verbe et la composition plate est obtenue par une entrée lexicale appropriée des verbes la déclenchant : lorsqu'un verbe V_1 admet un verbe V_2 comme complément, le système de ré-entrance de HPSG permet de faire monter les compléments de V_2 comme compléments de V_1.

Le formalisme des grammaires d'interaction ne permet pas cette montée des compléments et nous allons traiter les trois cas sur lesquels nous avons décidé de nous pencher, les auxiliaires de temps, les causatifs et les perceptifs, de trois façons différentes. Nous reprendrons pour l'essentiel une représentation déjà proposée pour les TAG mais nous verrons qu'elle est tout à fait compatible avec la représentation des clitiques que nous venons de présenter.

Les auxiliaires de temps

Nous traiterons les auxiliaires de temps comme des modifieurs de verbe qui prennent un participe passé et qui retournent un verbe à un temps composé et nous bloquerons la cliticisation du participe passé attendu en jouant sur la hiérarchie des catégories grammaticales $v_0, \ldots, v_6$. La figure 2.18 nous montre une entrée lexicale possible pour *ont* qui permet de réaliser ce mécanisme et,

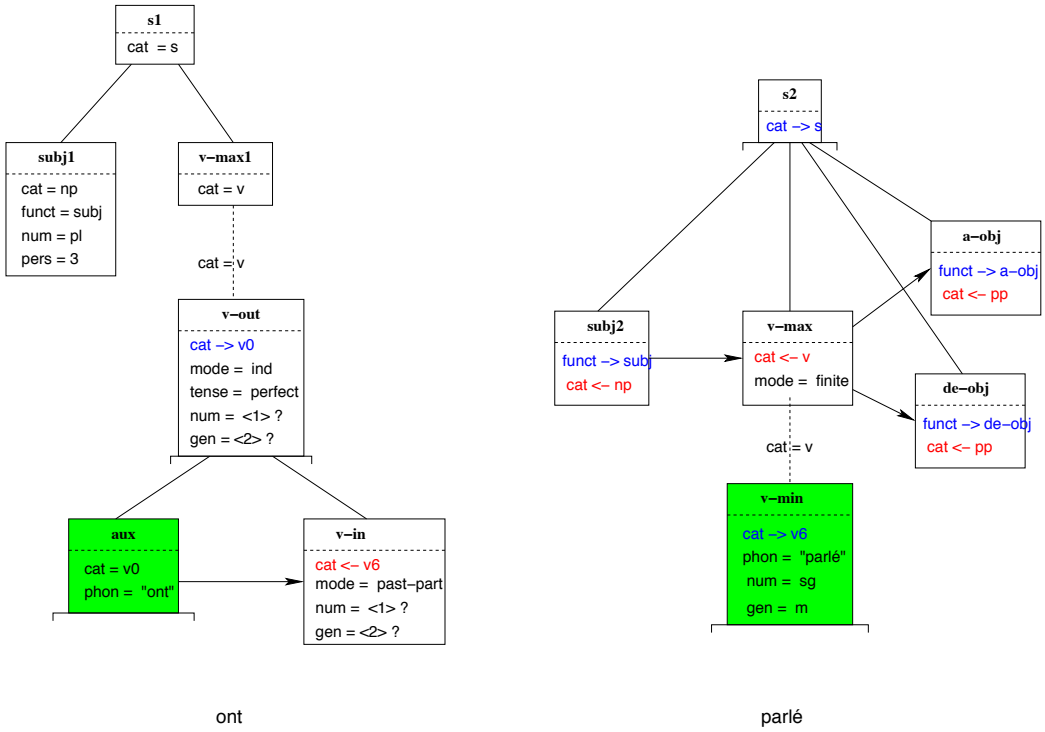


FIG. 2.18 – Descriptions syntaxiques fournies par un lexique pour *ont* et *parlé*

à côté, une entrée lexicale de *parlé* nous montre comment il pourrait être mis en œuvre. Il est facile de voir que ces entrées lexicales rendent possible la composition syntaxique de *« ont parlé »* et ensuite de *« ils lui en ont parlé »* alors que celle de *« * ils ont lui en parlé »* est impossible.

Cette modélisation est analogue à celle utilisée par A. Abeillé [Abe91] avec les TAG où les auxiliaires de temps sont associés à des arbres auxiliaires. M.-H. Candito [Can99] y a ajouté des traits particuliers *clS* et *clC* pour contrôler la présence des clitiques sujets et compléments.

Nous aurions pu adopter une composition plate comme dans HPSG mais nous n'aurions pas pu rattacher cette composition à l'entrée lexicale de l'auxiliaire comme dans HPSG car les grammaires d'interaction ne permettent pas changer la diathèse du participe passé par composition avec l'auxiliaire. Il aurait été nécessaire d'ajouter une entrée particulière du participe passé destinée à se composer avec un auxiliaire à la voix active. Avec le choix qui a été fait, nous pouvons utiliser une entrée lexicale du participe passé qui n'est pas structurellement différente de celle du verbe à un autre mode. Ainsi, cela permet de minimiser le nombre d'entrées du lexique.

Les verbes causatifs

Dans le cas de la « montée des clitiques »provoquée par un verbe causatif, il n'est pas possible de traiter le verbe causatif comme on a traité les auxiliaires de temps, car il est nécessaire de changer la diathèse du verbe causé. Par

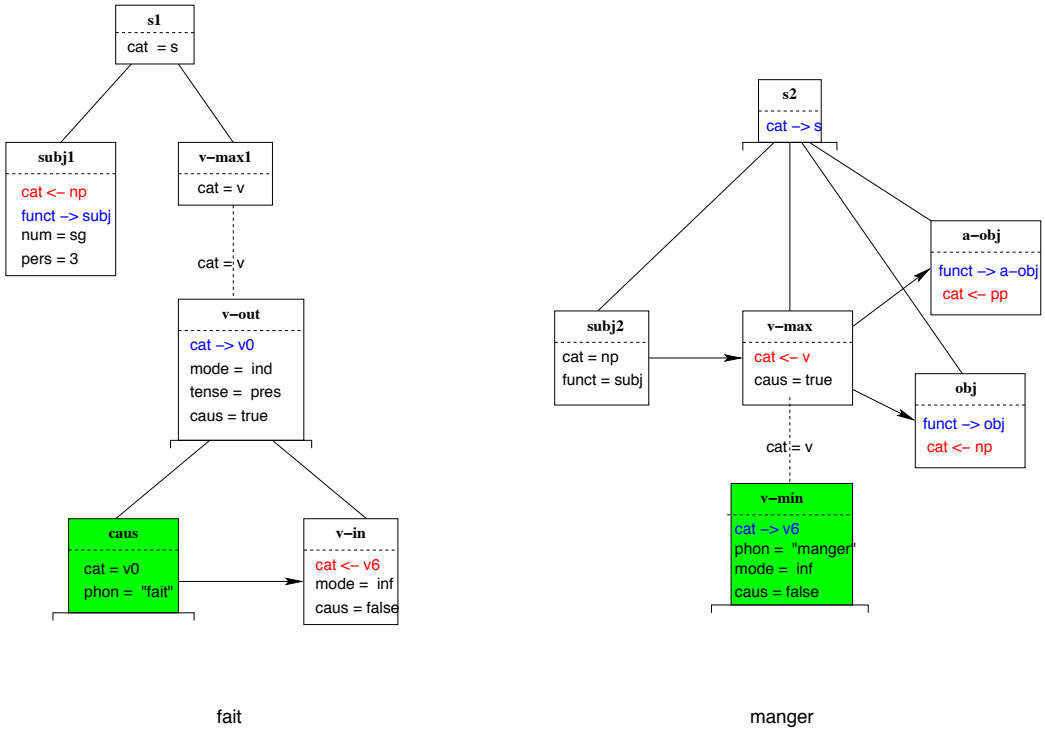


FIG. 2.19 – Descriptions syntaxiques fournies par un lexique pour *fait* et *manger* lorsqu'ils interviennent dans une construction causative

exemple, dans la phrase (b) *« Jean le lui fait manger »*, le verbe causatif est *fait* et le verbe causé est *manger*. Le sujet de *manger* devient l'objet indirect de *fait* exprimé par le pronom *lui*. Nous avons choisi de rattacher la construction causative à une entrée lexicale particulière du verbe causé de la même façon qu'il y a une entrée lexicale particulière pour la construction active et une autre pour la construction passive. Nous reprenons ainsi le choix effectué par M.-H. Candito pour les TAG [Can99].

La figure 2.19 nous montre l'entrée lexicale de *manger* exprimant son utilisation comme verbe causé dans une construction causative et, à côté, on trouvera une entrée lexicale pour le verbe causatif *fait*. La description syntaxique de *fait* exprime celui-ci comme un modifieur d'infinitif alors que celle de *manger* montre le changement de diathèse du verbe où le sujet devient complément d'objet indirect. On utilise de la même façon que pour les auxiliaires de temps les catégories v_0 et v_6 pour bloquer la cliticisation du verbe causé. Avec ces descriptions, il est possible d'analyser comme correct *« Jean le lui fait manger »* et de rejeter *« * Jean lui fait le manger »*. C'est un trait booléen *caus* qui impose la nécessité d'adjoindre un verbe causatif à l'infinitif. Avec cette représentation, ce qu'on appelle montée de clitiques n'en est pas vraiment une car *fait manger* se comporte comme un verbe unique.

Il est une exception à la montée des clitiques dans les constructions causatives, il s'agit des pronoms réfléchis. On dit par exemple *« Jean fait se laver Marie »* et non pas *« Jean se fait laver Marie »* si on veut que *se* réfère à *Marie*.

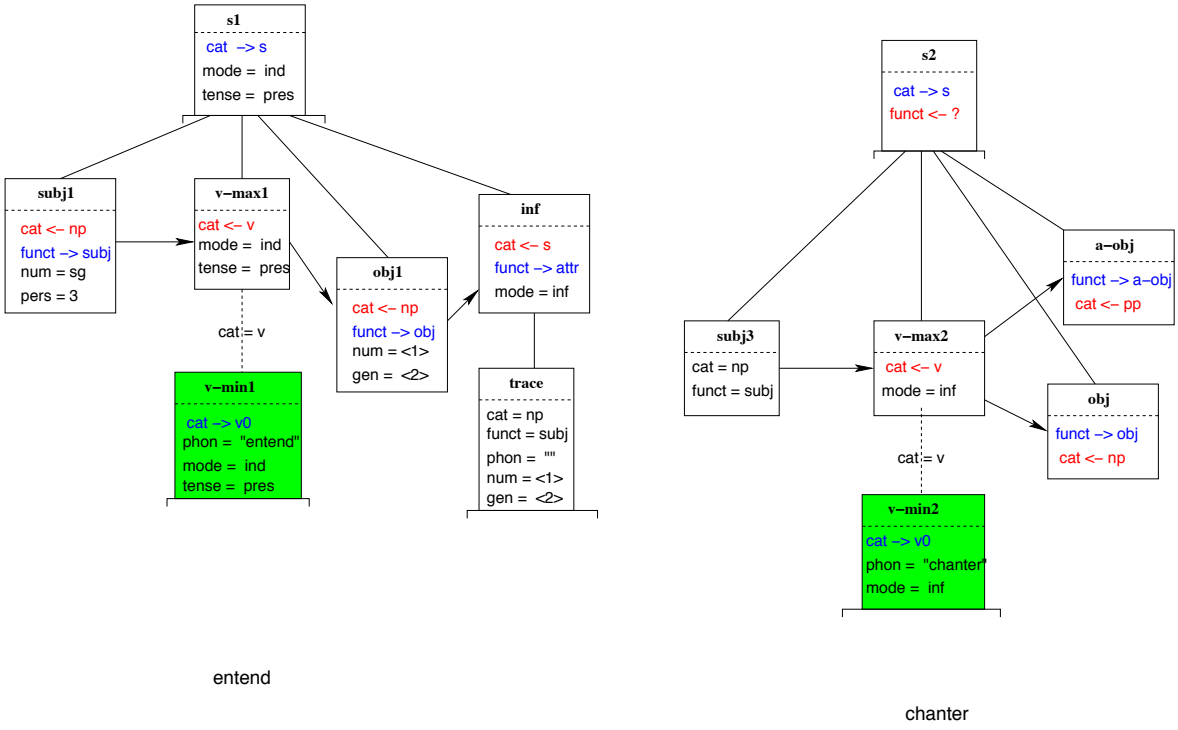


FIG. 2.20 – Descriptions syntaxiques fournies par un lexique pour *entend* et *chanter*

Cette exception est prise en compte en considérant une entrée lexicale particulière pour *laver* qui combine construction causative et forme pronominale.

Les verbes perceptifs

La meilleure façon de modéliser leur comportement est de les considérer comme des verbes ordinaires qui admettent un complément d'objet direct et un autre complément sous forme d'une infinitive qui a comme sujet l'objet direct précédent. On peut assimiler cette infinitive à un attribut de l'objet au même titre que les phrases suivantes contiennent des attributs de l'objet :

Jean trouve Pierre chantant.
Jean trouve Pierre épuisé.

Nous reprenons en cela le choix effectué par M.-H. Candito pour les TAG [Can99].

La figure 2.20 nous montre les descriptions syntaxiques que pourrait associer un lexique aux mots *entend* et *chanter*. Elles permettent d'accepter la phrase (d) *« Jean nous entend lui chanter une chanson »* et de rejeter aussi bien *« * Jean entend nous lui chanter une chanson »* que *« * Jean nous lui entend chanter une chanson »*. Encore une fois, il n' y a pas ici de montée de clitiques.

Pour prendre en compte des phrases comme *« Jean entend chanter une chanson par Marie »* ou *« Jean l'entend chanter par Marie »* , il faut envisager

une autre entrée lexicale pour *entend* du type de celle imaginée pour les verbes causatifs.

Pour conclure, il est important de rappeler que les exemples qui viennent d'être décrits ne sont là que pour illustrer le pouvoir expressif du formalisme. Les choix linguistiques sur lesquels ils reposent n'ont rien de définitif et on pourrait très bien envisager d'autres choix qui donnent d'autres modélisations.

Chapitre 3

L'analyse syntaxique électrostatique

Dans ce chapitre, nous allons nous intéresser aux méthodes d'analyse syntaxique utilisant les grammaires d'interaction [11].

Comme nous l'avons dit dans le chapitre précédent, l'analyse syntaxique se décompose en deux phases, une phase de sélection des descriptions syntaxiques associées aux mots de la phrase, appelée *étiquetage syntaxique*, suivie d'une phase d'analyse syntaxique proprement dite. Ces deux phases ont une complexité explosive. Pour la première, cela est dû à l'ambiguïté lexicale. En effet, la plupart des mots ont plusieurs entrées lexicales, par conséquent, le nombre d'étiquetages possibles d'une phrase est exponentiel en sa taille. Prenons par exemple la phrase *« la porte que la belle ferme présente ferme mal. »*[12]. Le mot *la* peut être un clitique, un déterminant ou encore la note de musique ! Dans le cas de *porte*, il peut s'agir du verbe transitif, du verbe intransitif ou du nom commun. Le mot *que* a trois entrées, *ferme* en a cinq, *présente* deux et *mal* deux également. Ceci nous donne donc $(3 \times 3 \times 3 \times 3 \times 3 \times 5 \times 2 \times 5 \times 2)$ c'est-à-dire 24300 étiquetages possibles, et ce, pour une phrase de moins de dix mots.

Une fois donné un étiquetage, l'analyse syntaxique proprement dite peut se résumer à associer les nœuds syntaxiques par paires pour réaliser les neutralisations de traits opposés. Dans notre exemple, il y a (au moins) 11 traits qui ont une polarité positive, qui attendent d'être neutralisés avec les traits correspondant de polarité négative. Ce qui nous donne une borne inférieure de $11! \simeq 4.10^7$ possibilités. En résumé, nous pouvons estimer à au moins $11! \times 24300$ le nombre de choix à faire, pour donner une idée, cela fait environ 10^{12} branches de calcul. En conclusion, la méthode qui viserait à résoudre le problème de manière directe, en testant toutes les possibilités n'a aucune chance de succès.

Si on le regarde d'un point de vue théorique, le problème général de l'analyse avec les grammaires d'interaction est NP-complet (l'entrée étant constituée à la fois de la grammaire et de la phrase à analyser). En effet, il est facile de montrer qu'il est NP et le problème de la dérivabilité dans le fragment des

[11] Ce travail a été mené en commun avec G. Bonfante et B. Guillaume.

[12] Cette phrase a été reprise légèrement modifiée de [Bla01].

clauses de Horn de logique linéaire, qui a été démontré NP-complet par M. Kanovitch [Kan92], peut être codé comme un problème de reconnaissance d'un langage engendré par une grammaire d'interaction. Mais notre but est d'analyser les langues naturelles et il s'agit d'appliquer des méthodes spécifiques à ce domaine. Nous allons montrer dans les deux sections suivantes comment l'utilisation des polarités va permettre d'éliminer de façon drastique des branches inutiles de l'arbre de calcul.

3.1 Étiquetage électrostatique

Dans cette section, nous nous intéressons à la première phase du processus d'analyse c'est-à-dire au problème de l'étiquetage. Il s'agit de retrouver pour chaque mot la description syntaxique qui lui correspond dans le lexique. Mais il y a une forte contrainte : il ne faut pas éliminer à cette étape d'étiquetage qui pourrait aboutir à une solution correcte. En effet, la deuxième phase de l'analyse ne pourra pas réparer les erreurs faites à ce stade. En conséquence de quoi nous préférons garder un ensemble de solutions plutôt qu'une seule fort probable mais pas sûre.

Ceci nous amène à écarter l'utilisation d'étiqueteurs standards comme les étiqueteurs de Brill ou encore les systèmes à base de chaînes de Markov cachées. D'autre part, les étiquettes correspondent dans notre cas aux descriptions syntaxiques du lexique, étiquettes qui contiennent plus d'informations que les étiquettes utilisées couramment. De ce point de vue, notre travail se rapproche plus du « super-tagging »de Joshi et Srivinas [JS94] ou encore de celui de Boullier [Bou03]. Les premiers visent à conserver les étiquetages les plus probables alors que Boullier n'utilise pas de probabilités et cherche à conserver tous les étiquetages qui mènent à des solutions. Nous nous situons aussi dans cette philosophie mais notre spécificité consiste à utiliser les polarités.

Le principe est de ne garder que les étiquetages globalement neutres. En effet, nous pouvons observer que le principe d'analyse garde invariant la somme des polarités d'un ensemble de nœuds. Comme les solutions ont des traits portant uniquement les polarités $=$ ou $\leftrightarrow$, la somme des polarités est neutre. En fait, c'est par nom de trait que l'on peut travailler ; pour chacun des noms, l'étiquetage, pour qu'il y ait une chance de succès, doit être globalement neutre. Notre méthode consiste à filtrer de manière rapide de tels étiquetages.

Il faut noter ici un point important, nous en avons fait la remarque précédemment, le nombre d'étiquetages possibles est exponentiel. Il est donc hors de question de considérer un à un chaque étiquetage. Pour ce faire, nous utilisons des automates. Un chemin dans l'automate correspond à un étiquetage, un état à un ensemble d'étiquetages.

3.1.1 Automates de polarité

En préalable à la construction des automates, il est nécessaire de définir précisément une fonction qui, pour une valeur atomique v de trait, additionne les polarités de v dans une description syntaxique donnée.

Définition 3.1.1. *Étant donnée une description syntaxique D, un trait f, une*

valeur atomique de trait $v \in \mathcal{V}_f$ et un nœud $N \in D$, on définit

$$p_N(v) = \begin{cases} \{0\} & si\ v \notin v_f(N) \\ \{1\} & si\ v = v_f(N)\ et\ p_f(N) = \rightarrow \\ \{-1\} & si\ v = v_f(N)\ et\ p_f(N) = \leftarrow \\ \{0,1\} & si\ v < v_f(N)\ et\ p_f(N) = \rightarrow \\ \{0,-1\} & si\ v < v_f(N)\ et\ p_f(N) = \leftarrow \end{cases}$$

Étant donnée une description syntaxique D dont les nœuds sont $N_1, N_2, \ldots, N_k$, on définit $p_D(v) = \{n \in \mathbb{Z} \mid n = n_1 + n_2 + \cdots + n_k$ avec $n_1 \in p_{N_1}(v), n_2 \in p_{N_2}(v), \ldots, n_k \in p_{N_k}(v)\}$.

Remarque 3.1.1. *La fonction p_D retourne un ensemble d'entiers et non un entier unique du fait de l'indéterminisme lié à la présence de valeurs disjonctives dans les traits de D. On peut même dire que cet ensemble est toujours un segment de $\mathbb{Z}$.*

Nous supposons pour le reste de la section que nous avons une phrase $w_1 w_2 \ldots w_n$, et que pour chaque $1 \leq i \leq n$ le mot w_i contient dans le lexique les descriptions syntaxiques $D_{i,1}, D_{i,2} \ldots D_{i,k_i}$.

Pour chaque valeur atomique v, on construit l'automate A_v suivant. L'ensemble des états de A_v est $[0..n] \times \mathbb{Z}$ où n est la longueur de la phrase. L'état initial est $(0,0)$, l'état final (unique) est $(n,0)$. Les transitions sont étiquetées par les descriptions $D_{i,j}$. Étant donné une telle description $D_{i,j}$, il y a une transition entre $(i-1,x) \xrightarrow{D_{i,j}} (i,y)$ si et seulement s'il existe un entier $z \in p_{D_{i,j}}(v)$ tel que $y = z + x$.

Atteindre l'état (i,t) depuis l'état initial $(0,0)$ signifie alors que

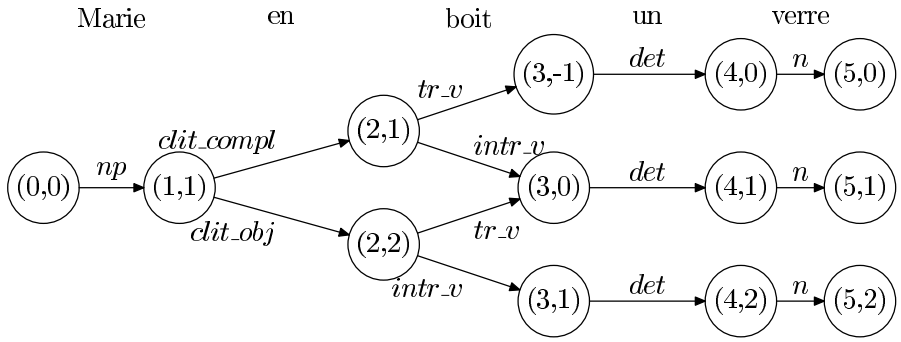
(a) le chemin emprunté a la forme $D_{1,j_1}, D_{2,j_2}, \ldots, D_{i,j_i}$, c'est-à-dire consiste en un étiquetage des i premiers mots,

(b) t est une valeur possible pour la somme des polarités correspondant à la valeur v le long de ce chemin.

En conséquence de quoi, tout chemin menant à l'état final correspond intuitivement à un étiquetage globalement neutre pour cette valeur v.

Remarque 3.1.2. *Observons que l'automate A_v n'est pas forcément déterministe. En effet, dès qu'une description $D_{i,j}$ contient un trait polarisé (non neutre) dans lequel la valeur v est au sein d'une disjonction, la somme $p_{D_{i,j}}(v)$ est un ensemble contenant au moins deux éléments. Il y aura dans ce cas autant de transitions étiquetées par $D_{i,j}$ qu'il n'y a d'éléments dans $p_{D_{i,j}}(v)$.*

L'automate ci-dessous est l'automate A_{np} pour la phrase *« Marie en boit un verre »*. Dans notre lexique, nous avons une entrée pour *Marie* (nom propre), deux entrées pour *en* (clitique objet — *Marie en boit* – et, pour le cas présent, clitique complément de nom), deux entrées pour *boit* (verbe transitif, intransitif), une pour *un* (déterminant), une pour *verre* (nom commun).



Dans l'exemple, il y a quatre étiquetages possibles, un seul chemin aboutit à l'état final (5,0), il s'agit de l'étiquetage correct.

On peut construire ainsi pour chacune des valeurs de trait v, un automate A_v ; chaque automate filtre certains étiquetages. Il ne reste plus qu'à en prendre l'intersection. Cela revient à faire l'intersection des automates A_v. Nous calculons alors l'automate $A = \bigcap_{v \in V} A_v$ qui ne retient que les chemins (susceptibles d'être) neutres. Le critère que nous proposons voit la phrase globalement, il ne tient pas compte des contraintes locales. En particulier, si un étiquetage est correct, toute permutation de cet étiquetage compatible avec le lexique le sera. Par exemple, dans *« la fille la voit »*, l'étiquetage $det, n, clit, trans_v$ est correct, mais $clit, n, det, trans_v$ le sera par conséquent.

Pour pallier ce défaut, nous ajoutons un filtre qui supprime des séquences interdites d'étiquettes. Ce filtre tient compte des réalités linguistiques, par exemple, on ne peut pas avoir un « adjectif gauche »suivi directement d'un « adjectif droit » ; il faut un nom commun entre les deux. On ne peut pas garantir que ce nouveau filtre n'écarte pas des étiquetages menant à des solutions car il repose sur l'expertise linguistique de celui qui écrit les séquences interdites. L'intérêt du procédé est qu'il fonctionne comme les autres : les séquences interdites sont des expressions régulières, et par conséquent, le filtrage correspond à une ultime intersection avec l'automate obtenu auparavant. Dans l'exemple *« la fille la voit »*, le filtre supprime l'étiquetage qui attribue au second *la* la catégorie de déterminant et à *voit* celle de verbe transitif ; en effet, un déterminant ne peut être suivi immédiatement par un verbe transitif.

3.1.2 Estimation de l'efficacité du filtrage

Pour estimer l'efficacité du filtrage, nous pouvons employer un argument probabiliste. Le problème de l'étiquetage se rapporte en fait à une question de marche aléatoire. L'automate A issu du filtrage électrostatique a théoriquement ses états dans $([0..n] \times \mathbb{Z})^d$ où d est le nombre de valeurs de traits ayant servi à faire le filtrage car il est le résultat de l'intersection de d automates dont les états sont dans $[0..n] \times \mathbb{Z}$. Mais en fait, les état de A sont tous de la forme $((i,p_{i_1}),\ldots,(i,p_{i_d}))$, ce qui signifie qu'on lit les i premiers mots de la phrase dans les d automates en parallèle. On peut donc simplifier les états de A pour les écrire $(i,p_{i_1},\ldots,p_{i_d})$. Prenons maintenant un chemin dans cet automate. Si l'on oublie la première composante des états, on se déplace dans $\mathbb{Z}^d$: un pas correspond à une transition étiquetée par une description D et le déplacement

selon une direction v_i va être donné par le nombre de polarités de v_i contenues dans D.

Le modèle de marche aléatoire que l'on considère est le suivant. Pour un vecteur $\mathbf{v} \in \mathbb{Z}^d$, on note $\pi_{\mathbf{v}}$ la probabilité qu'un mot ait une description D qui vérifie l'égalité : $(p_{v_1}(D), p_{v_2}(D), \ldots, p_{v_d}(D))) = \mathbf{v}$ où $v_1, v_2, \ldots, v_d$ sont les valeurs de traits sur lesquels on fait le filtrage. En d'autres termes, c'est la probabilité que dans l'automate (en oubliant la première dimension) on se déplace d'un pas suivant $\mathbf{v}$.

La probabilité qui nous intéresse, c'est la probabilité que l'on atteigne le point $\mathbf{O} = (0, 0, \ldots, 0)$ à partir de ce même point $\mathbf{O}$ à la fin de la phrase. Un tel chemin correspond en effet à un chemin retenu par le filtre. Notons P_n la probabilité que l'on atteigne $\mathbf{O}$ partant de $\mathbf{O}$ après n pas. Comme les probabilités sont arbitraires, on peut supposer sans risques que le pas moyen est non nul, $\sum_{\mathbf{v} \in \mathbb{Z}^d} \pi_{\mathbf{v}} \mathbf{v} \neq \mathbf{0}$. Dans ce contexte, c'est un résultat classique, on peut borner la probabilité $P_n \leq \beta^n$ pour un certain $\beta < 1$, et n assez grand. En d'autres termes, plus la phrase est longue, plus la proportion d'étiquetages conservés est petite, et même, cette proportion est inversement exponentielle à la longueur de la phrase.

3.2 Analyse syntaxique contrôlée

Dans la suite, quand nous parlerons de *modèles*, il sera sous-entendu qu'il s'agit de modèles neutres et minimaux tels qu'ils sont définis dans la section 2.2. Dans toute cette section, nous ne nous intéressons qu'à la seconde phase de l'analyse syntaxique. Pour une phrase $w_1, w_2, \ldots w_n$, on suppose que l'on a choisi dans le lexique une description D_i pour chacun des mots w_i. On appellera description initiale la description D_0 définie comme la conjonction des D_i, avec des contraintes de préséance supplémentaires pour tenir compte de l'ordre des mots de la phrase. Le but de la seconde phase de l'analyse syntaxique est de trouver les modèles de cette description D_0.

Dans une première étape, pour construire ces modèles, on utilise les opérations élémentaires de neutralisation de traits polarisés et de simplification. Comme les opérations de simplification ne changent pas l'ensemble des modèles, on les applique systématiquement après chaque neutralisation.

Pour éviter l'explosion combinatoire due au nombre de possibilités de neutraliser les traits opposés, on va guider la recherche de deux façons. D'une part, on impose un ordre sur les neutralisations de traits pour éviter de dupliquer les solutions. D'autre part, on analyse la phrase de gauche à droite en bornant, à chaque étape, pour la partie de la phrase déjà analysée, le nombre de nœuds portant encore des polarités.

Après cette première étape, on obtient des descriptions neutralisées. Il reste alors à construire les modèles de ces descriptions en réalisant les relations de domination large restantes. En théorie, cette étape peut être coûteuse en temps de calcul. Cependant, si la grammaire introduit suffisamment de contraintes, la majeure partie des fusions de nœuds a été forcée par les polarités.

3.2.1 Un ordre sur les neutralisations

Pour un modèle donné de la description initiale D_0, on peut définir l'ensemble des couples de traits opposés qui sont neutralisés. Toutes les suites de couples de traits actifs qui sont des permutations de cet ensemble vont conduire à la même solution. Si k couples de traits actifs sont à neutraliser, en utilisant la programmation dynamique, il y a C_k^i choix de i couples et, pour chacun de ces couples, il y a $k-i$ choix possibles d'un nouveau couple à neutraliser. Donc au total le nombre de neutralisations est de $\sum_{i=1}^{k}(k-i)C_k^i = k2^{k-1} = O(2^k)$. Pour éviter de dupliquer les solutions, nous n'allons chercher que les suites respectant un ordre fixé, suites que nous appellerons suites canoniques. On assure ainsi que nous n'explorons qu'une fois chaque ensemble de couples de traits opposés. Pour définir ces suites canoniques, il va falloir définir un ordre sur les couples de traits neutralisables et, pour cela, il faut commencer par définir un ordre sur les traits susceptibles de participer à ces couples. Il s'agit des traits positifs et négatifs que nous qualifierons globalement de *traits actifs*. Cet ordre met en jeu les nœuds auxquels sont attachés ces traits d'où la nécessité de les identifier soigneusement.

Définition 3.2.1. *Un* trait actif *d'une description D est un couple (N, f) composé d'un nom N de nœud de D et d'un nom f tel que le trait f porte une polarité positive ou négative dans N.*

Remarque 3.2.1. *Il est essentiel dans la suite de distinguer les traits actifs (N, f) et (N', f) même si N et N' sont deux noms du même nœud. On va totalement ordonner les traits actifs et on veut que cet ordre soit stable par neutralisation et simplification. Dans la suite on suppose donc qu'un trait actif est toujours désigné par le nom qu'il portait dans la description initiale D_0.*

Comme on simplifie toujours au maximum après une neutralisation, on utilise la notation $D \overset{N\overset{f}{=}M}{\Longrightarrow} D''$ pour dire que $D \overset{N\overset{f}{=}M}{\longrightarrow} D'$ et que D'' est la simplification maximale de D'.

Pour une description initiale D_0 fixée et TA_0 l'ensemble de ses traits actifs, on appelle *suite de neutralisations* une suite $(N_i, M_i, f_i)_{0\leq i\leq k}$ telle qu'il existe une suite $(D_i)_{1\leq i\leq k+1}$ de descriptions et une suite $(TA_i)_{1\leq i\leq k+1}$ vérifiant :

- (N_i, f_i) et (M_i, f_i) sont dans TA_i.
- $D_i \overset{N_i\overset{f_i}{=}M_i}{\Longrightarrow} D_{i+1}$
- TA_{i+1} est l'ensemble des traits de TA_i encore actifs dans D_{i+1}.

On fixe un ordre total sur l'ensemble des nœuds de D_0 compatible avec l'ordre des mots de la phrase (i.e. un ordre tel que, si $i < j$ alors tout nœud de la description D_i associée à w_i précède tout nœud de la description D_j associée à w_j). Si, de plus on fixe un ordre total arbitraire sur $\mathcal{F}$, alors TA_0 est totalement ordonné par l'ordre lexicographique $\prec$:

$$(N, f) \prec (N', f') \iff N < N' \text{ ou } (N = N' \text{ et } f < f')$$

On ordonne alors les couples de traits actifs avec la relation $\ll$ définie par :

$((N_1, f_1), (M_1, f_1)) \ll ((N_2, f_2), (M_2, f_2)) \iff max((N_1, f_1), (M_1, f_1)) \prec max((N_2, f_2), (M_2, f_2))$

On note $D \overset{N\overset{f}{=\!=\!>}M}{\Longrightarrow} D'$ si $D \overset{N\overset{f}{=\!=\!>}M}{\Longrightarrow} D'$ et $((N,f),(M,f))$ est l'élément minimal de l'ensemble des couples de traits neutralisés durant les opérations de neutralisation et de simplification.

La suite de neutralisations $(N_i, M_i, f_i)_{0\leq i\leq k}$ est canonique si :

– pour tout i, $D_i \overset{N_i\overset{f_i}{=\!=\!>}M_i}{\Longrightarrow} D_{i+1}$;

– la suite des $((N_i,f_i),(M_i,f_i))_{0\leq i\leq k}$ est $\ll$-strictement croissante.

Avec cette définition de suite canoniques, on peut prouver la proposition suivante qui assure qu'il suffit de neutraliser les traits suivant l'ordre canonique pour, d'une part, trouver toutes les solutions et, d'autre part, ne les trouver qu'une fois chacune.

Proposition 3.2.1. *Soit D une description et T un modèle de D. Alors, il existe une unique suite canonique de neutralisations de D à D' telle que D' soit une description neutralisée admettant T comme modèle.*

3.2.2 Une borne sur les traits actifs

Malgré l'ordre défini précédemment, la taille de l'arbre de recherche peut rester très grande. En s'appuyant sur une heuristique d'ordre psycholinguistique, on va privilégier certaines branches de cet arbre de recherche pour améliorer les performances de l'algorithme. Lorsqu'un humain lit une phrase, il procède de façon incrémentale, du début à la fin de celle-ci. On peut supposer que, durant ce processus, il utilise une mémoire à court terme de petite taille pour stocker les informations nécessaires à l'analyse syntaxique. Dans notre contexte, cela va se traduire par une analyse procédant de façon incrémentale : la description initiale D_0 est pour l'essentiel une juxtaposition des descriptions $D_1, \ldots, D_n$ associées aux mots $w_1, \ldots, w_n$ de la phrase. Ces descriptions vont être parcourues dans cet ordre et leurs nœuds actifs, ceux qui sont porteurs de traits actifs, vont être mis dans une mémoire dont la taille est fixée à l'avance et dès que la mémoire est pleine, il est nécessaire de procéder à des neutralisations pour pouvoir continuer l'analyse. Plus la taille de la mémoire est petite, plus les neutralisations vont être forcées rapidement et plus l'analyse sera efficace mais en même temps on risque de manquer certaines analyses.

Pour définir formellement l'algorithme, il est nécessaire d'introduire la notion de potentiel d'une description syntaxique relatif à une position.

Définition 3.2.2. *Notons $\mathcal{N}_i$ l'ensemble des noms de nœuds de la description D_i du lexique associée à w_i et $\mathcal{N}_{\leq k} = \bigcup_{1\leq i\leq k} \mathcal{N}_i$.*

Soit D une description définie sur le même ensemble de noms de nœud que la description initiale D_0 . Le potentiel $pot_k(D)$ de D relatif à la position k est le nombre de nœuds actifs de D présents dans l'ensemble de nœuds $\mathcal{N}_{\leq k}$.

Soit $b \in \mathbb{N}$ une borne fixée sur le nombre de nœuds actifs. L'algorithme agit sur un ensemble d'états qui sont représentés par des triplets de la forme (D, k, p) où :

– D représente l'état courant de la description syntaxique associée à la phrase analysée ;

– k représente le nombre de mots de la phrase déjà considérés ;

– p représente le plus grand des deux traits actifs utilisés pour la dernière neutralisation (au sens de l'ordre défini sur les traits actifs).

L'algorithme se présente sous forme de deux règles de transition permettant de remplacer un ancien état par un ensemble de nouveaux :

– **Shift :** si $pot_k(D) \leq b$ et $k < n$ alors $(D, k, p) \rightsquigarrow \{(D, k+1, p)\}$;

– **Reduce :** sinon $(D, k, p) \rightsquigarrow \{(D', k, p') \mid N_1, N_2 \in \mathcal{N}_{\leq k}$ et $D \overset{N_1 \overset{f}{=\!\!\Rightarrow} N_2}{\Longrightarrow} D'$ et $p \prec p' = \max((N_1, f), (N_2, f))\}$.

L'algorithme est initialisé avec l'ensemble $\{(D_0, 0, \bot)\}$ (où $\bot$ est une constante inférieure à tous les traits actifs) et se termine quand on atteint un point fixe. Les solutions vont être les modèles des descriptions D des états de la forme (D, n, p) où D est une description neutralisée, n le nombre de mots de la phrase et p quelconque.

3.2.3 Recherche des modèles des descriptions neutralisées

En théorie cette recherche peut être coûteuse. En effet, si on ne polarise aucun trait dans les descriptions associées aux mots de la phrase, toute l'analyse se fait ici. Cependant, tout l'intérêt des grammaires d'interaction est de guider l'analyse grâce aux polarités. Si les entrées de la grammaire sont suffisamment polarisées, cette recherche est très rapide. En pratique, il reste pas ou très peu de relations d'ascendance dans les descriptions neutralisées.

Pour simplifier la recherche de modèles, on suppose que toutes les descriptions neutralisées obtenues avec l'algorithme précédent sont connexes (pour l'union des relations de parenté et d'ascendance). Il est en effet facile de s'apercevoir que l'absence de connexité peut entraîner la multiplication des modèles.

Dans le cas de la grammaire que nous avons implémentée (voir section suivante), toutes les descriptions élémentaires (sauf celle du point final) sont de l'une des deux formes suivantes :

– soit il y a un seul trait positif *cat* ;

– soit il y a un trait positif *cat* dans un nœud qui est un ascendant strict d'un nœud avec un trait négatif *cat*.

La seule description qui ne porte qu'un trait négatif *cat* est celle associée au point final. En supposant qu'il n'y a toujours exactement un point final, toutes les descriptions neutralisées obtenues sont connexes.

Une fois admise la connexité, la source de multiplicité des modèles est dans les relations d'ascendance non encore réalisées. Nous ne détaillerons pas l'algorithme qui permet de les éliminer mais le principe en est le suivant : s'il reste par exemple une relation $N \overset{\star}{>}_S N'$, on va utiliser le père éventuel de N' et les fils éventuels de N pour réduire la relation d'ascendance.

3.3 Implémentation et résultats

Les grammaires d'interaction sont implémentées dans un premier prototype baptisé LEOPAR[13] (eLEctrOstatic PARser). Une partie des idées présentées dans ce mémoire sont nées de l'expérience acquise durant le développement de ce prototype et ne sont pas encore implémentées. Par exemple, LEOPAR

[13] voir `www.loria.fr/equipes/calligramme/leopar`

ne permet pas de coder la relation $N > \{N_1, \ldots, N_k\}$. Nous nous sommes donné certaines contraintes quant au choix de représentation des données : par exemple, la structure de données utilisée pour coder les descriptions d'arbres n'autorise qu'un ancêtre au plus par nœud et les règles de simplification sont donc adaptées pour tenir compte de cette contrainte. Ce prototype est réalisé en Objective CAML (5000 lignes de code). Il permet à l'utilisateur de paramétrer les noms et valeurs de traits, le lexique et la grammaire utilisés par l'analyseur. Il permet enfin de choisir une borne sur le nombre de nœuds actifs.

La distribution actuelle de LEOPAR est accompagnée d'un petit lexique et d'une grammaire qui comportent respectivement 78 formes fléchies et 31 descriptions syntaxiques. Les fichiers du lexique et de la grammaire sont consultables sur Internet à partir de la page du logiciel.

La souplesse des grammaires d'interaction permet de couvrir bon nombre de phénomènes linguistiques avec un nombre restreint de descriptions de la grammaire. Dans le tableau 3.1, nous donnons quelques exemples de résultats obtenus avec notre prototype [14]. Les phénomènes traités par la grammaire

phrase analysée	borne min[i]	temps borne min/6[ii]	étiquetages conservés[iii]
(a_1) les souris que Jean trouve dorment.	4	0.02s / 0.07s	2 / 24
(a_2) Jean trouve les souris qui dorment.	4	0.01s / 0.04s	1 / 24
(b_1) Jean ne boit pas un verre.	4	0.02s / 0.06s	2 / 6
(b_2) Jean ne boit aucun verre.	4	0.01s / 0.03s	2 / 6
(b_3) aucune fille ne boit un verre.	3	0.01s / 0.05s	2 / 6
(b_4) Jean ne boit dans aucun verre.	4	0.01s / 0.04s	1 / 12
(b_5) Jean ne boit dans le verre d'aucune fille.	5	0.04s / 0.07s	1 / 48
(c_1) les souris que Jean pense que la boite contient dorment.	5	0.04s / 0.06s	1 / 216
(c_2) *les souris que Jean trouve la boite qui contient dorment.	—	— / 0.24s	2 / 432
(d_1) Jean à la fille de qui Marie présente les souris boit un verre.	5	0.19s / 1.19s	4 / 432
(d_2) Jean dans la ferme de qui Marie dort boit un verre.	6	2.26s / 2.26s	8 / 540
(e_1) la porte que la belle ferme présente ferme mal.	6	0.24s / 0.24s	6 / 64800
(e_2) la belle porte que la fille trouve que la ferme présente ferme mal.	5	0.50s / 0.69s	8 / 2332800
(e_3) Jean qui ferme une grande boite qui contient une souris pense que Marie dort dans une ferme que Jean présente.	4	0.16s / 6.65s	3 / 129600

i. plus petite borne qui permet d'obtenir toutes les analyses correctes

ii. temps d'analyse avec la borne indiqué dans la colonne précédente / temps d'analyse avec la borne 6.

iii. nombre d'étiquetages conservés par le super-tagging par rapport au nombre initial d'étiquetages possibles

TAB. 3.1 – Quelques résultats obtenus avec le prototype

[14] La machine utilisée est un P4, 1.7Ghz.

fournie avec LEOPAR sont entre autres, (a) les relatives, (b) les négations quelle que soit la position de *aucun*, (c) les barrières à l'extraction, (d) les dépendances non bornées en cascade dans les relatives. Le formalisme des grammaires d'interaction permet également de traiter de façon naturelle d'autres phénomènes qui n'ont pas encore été intégrés à l'implantation actuelle comme, par exemple, l'inversion du sujet dans les relatives ou l'ordre des clitiques. Les trois phrases (e) donnent une idée des performances de l'analyseur et de la phase d'étiquetage. (e_2) est volontairement très ambiguë. (e_3) comporte 20 mots mais peut être analysée rapidement.

Chapitre 4

L'intégration de la sémantique : les grammaires d'interaction synchrones

La syntaxe n'étant qu'un moyen pour accéder à la sémantique, un formalisme linguistique, même s'il se cantonne à la première, ne peut pas ignorer la seconde. Les grammaires d'interaction se situent dans le prolongement des grammaires catégorielles. Or, ces dernières permettent une intégration particulière de la sémantique sous forme de la sémantique de Montague [Car98]. Nous verrons comment la rigidité du lien entre syntaxe et sémantique dans les grammaires catégorielles entraîne un alourdissement considérable du formalisme quand on veut rendre compte de certains phénomènes sémantiques.

Nous verrons ensuite comment les grammaires d'interaction peuvent intégrer la sémantique d'une manière beaucoup plus souple pour former ce que nous appellerons les *grammaires d'interaction synchrones*. Nous y trouvons toujours deux niveaux distincts, celui de la syntaxe et celui de la sémantique. Le niveau sémantique est formalisé de façon semblable au niveau syntaxique par un recours aux polarités et aux descriptions de structures sous-spécifiées mais ces descriptions ne sont plus des descriptions d'arbres : ce sont des descriptions de graphes acycliques orientés (DAG pour Directed Acyclic Graphs). La synchronisation entre les deux niveaux se fait simplement par une fonction partielle projetant les nœuds syntaxiques sur des nœuds sémantiques.

Nous illustrerons la façon dont la sémantique interagit avec la syntaxe dans le formalisme des grammaires d'interaction synchrones par un certain nombre d'exemples significatifs. Enfin, nous situerons notre modélisation de la sémantique et de l'interface syntaxe-sémantique par rapport aux travaux de référence du domaine les plus proches.

4.1 La rigidité du lien entre syntaxe et sémantique dans les grammaires catégorielles

Considérons une grammaire catégorielle standard présentée sous forme d'une grammaire de Lambek. La sémantique y est intégrée à la Montague en intro-

duisant un ensemble de types sémantiques construits à partir des deux types de base *e (entités)* et *t (truth-values)* à l'aide de l'opérateur $\rightarrow$. A ces types, sont associés des λ-termes d'ordre supérieur construits à partir de variables et de constantes. Ces λ-termes vont servir à exprimer le sens des expressions de la langue et c'est pourquoi nous parlerons de *termes sémantiques* pour les désigner.

Le lien entre syntaxe et sémantique est réalisé tout d'abord au niveau des types à l'aide d'un homomorphisme qui projette tout type syntaxique atomique en un type sémantique (pas forcément atomique). Habituellement, on associe les types sémantiques t, e, $e \rightarrow t$ aux types syntaxiques de base S, NP et N. Cela veut dire qu'une phrase (S) est une proposition logique vraie ou fausse (t), un syntagme nominal (NP) est un individu (e) et un nom commun (N) une propriété définie par sa fonction caractéristique $e \rightarrow t$ qui indique si un individu quelconque a ou pas cette propriété. Ensuite, on projette un type syntaxique composé $A\backslash B$ ou B/A sur le type $\alpha \rightarrow \beta$ où α et β sont les projections sémantiques respectives de A et B.

Il reste enfin à établir un lien au niveau des termes. Si l'on considère les syntagmes comme des termes phonologiques typés dont les types sont des formules du calcul de Lambek, on peut aussi considérer leur représentation sémantique comme des λ-termes typés du langage de termes sémantiques qui vient d'être présenté. La correspondance entre termes phonologiques et termes sémantiques s'effectue à partir d'un lexique. Un lexique syntaxique d'une grammaire de Lambek associe des mots à des types syntaxiques. Pour le compléter en un lexique sémantique, il suffit en plus d'associer aux mots des termes sémantiques représentant leur sens. Une entrée lexicale se présentera donc maintenant sous la forme $w : T - \alpha : \tau$ où w est un mot de type syntaxique T dont le sens est exprimé par le terme sémantique α de type τ. Bien entendu, pour assurer la cohérence de l'entrée, il est nécessaire que τ soit la projection sémantique de T.

On a vu dans la section 1.3 que la composition syntaxique d'une phrase revenait à la construction d'un réseau de démonstration. La construction de la représentation sémantique de la phrase s'effectue en parallèle et la meilleure façon de l'exprimer est d'utiliser un étiquetage des réseaux de démonstration à la manière dont nous avons pu le faire dans la section 1.3. Dans cette section, nous avons vu qu'il était possible d'étiqueter un réseau par des termes phonologiques typés. Nous allons maintenant ajouter un deuxième étiquetage des réseaux à l'aide de termes sémantiques typés. Les règles d'étiquetage sont données par la figure 4.1. Les étiquettes phonologiques sont en gras et les étiquettes sémantiques en italiques. Les règles montrent qu'au niveau sémantique il n'y a plus de distinction entre implications gauche et droite, ce qui fait que la structure de dépendance sémantique de la phrase n'est que la projection de la structure syntaxique du niveau des réseaux de démonstration non commutatifs du calcul de Lambek au niveau commutatif des réseaux intuitionnistes. Dans cette remarque, réside toutes les limites des grammaires catégorielles comme nous le verrons bientôt.

Nous avons expliqué dans la section 1.3 comment la construction de réseaux syntaxiques pouvait se ramener à un problème d'unification de termes phonologiques. Voyons maintenant à travers un exemple classique comment

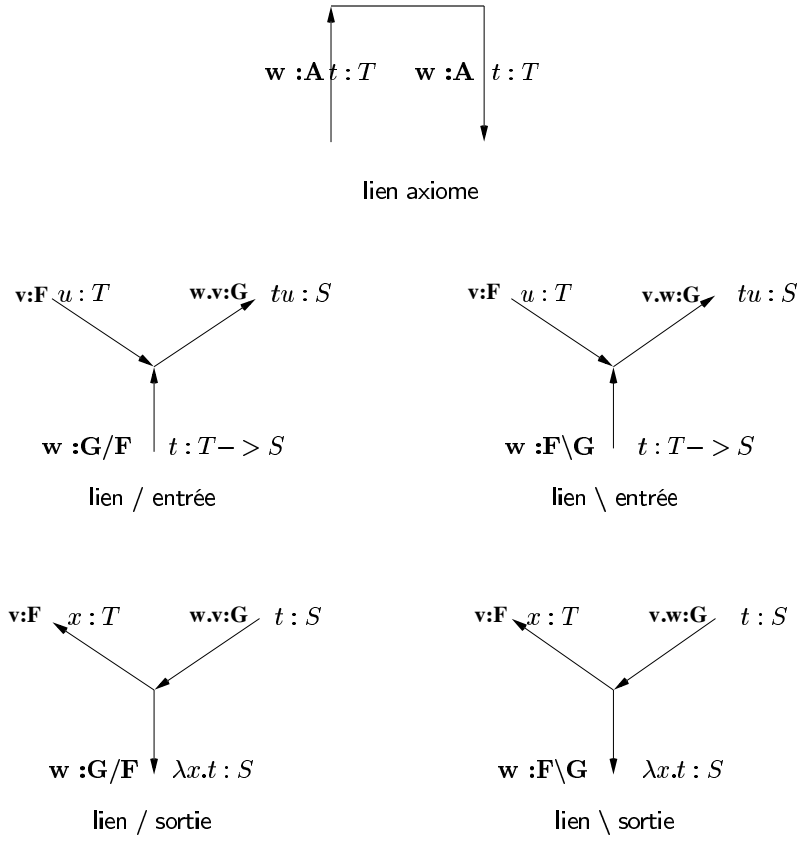


FIG. 4.1 – Règles d'étiquetage phonologique et sémantique d'un réseau de Lambek

la construction de la représentation sémantique de la phrase peut se ramener à un problème d'unification de termes sémantiques. Considérons la phrase *« tous aiment quelqu'un »* qui a deux lectures possibles selon l'ordre de priorité de portée entre les quantificateurs. On suppose qu'une grammaire catégorielle fournit les entrées lexicales suivantes pour les mots de la phrase :

$$\begin{array}{l}\textbf{tous}: \ \mathbf{S/(NP\backslash S)} - \lambda p.\forall u.pu: \ (e \to t) \to t \\ \textbf{aiment}: \ \mathbf{(NP\backslash S)/NP} - aimer: e \to (e \to t) \\ \textbf{quelqu'un}: \ \mathbf{(S/NP)\backslash S)} - \lambda q.\exists v.qv: \ (e \to t) \to t\end{array}$$

Le lecteur est peut-être surpris de ne pas trouver NP comme type syntaxique de *tous* et *quelqu'un*. La raison en est qu'au niveau sémantique on veut indiquer que ces quantificateurs ont une portée qui est l'ensemble de la phrase, ce qu'expriment les termes sémantiques associés à ces mots par le lexique. Ces termes ont tous les deux le type $(e \to t) \to t$. Ce type doit être la projection du type syntaxique des deux mots donc cela ne peut pas être NP si l'on considère que NP se projette en e. On a alors deux possibilités : soit modifier le type sémantique associé à NP, soit choisir pour les mots *tous* et *quelqu'un* un type syntaxique compatible avec leur type sémantique. En supposant que notre lexique soit beaucoup plus conséquent que ce que les 3 entrées présentées laissent supposer, la première option obligerait à revoir toute la cohérence de

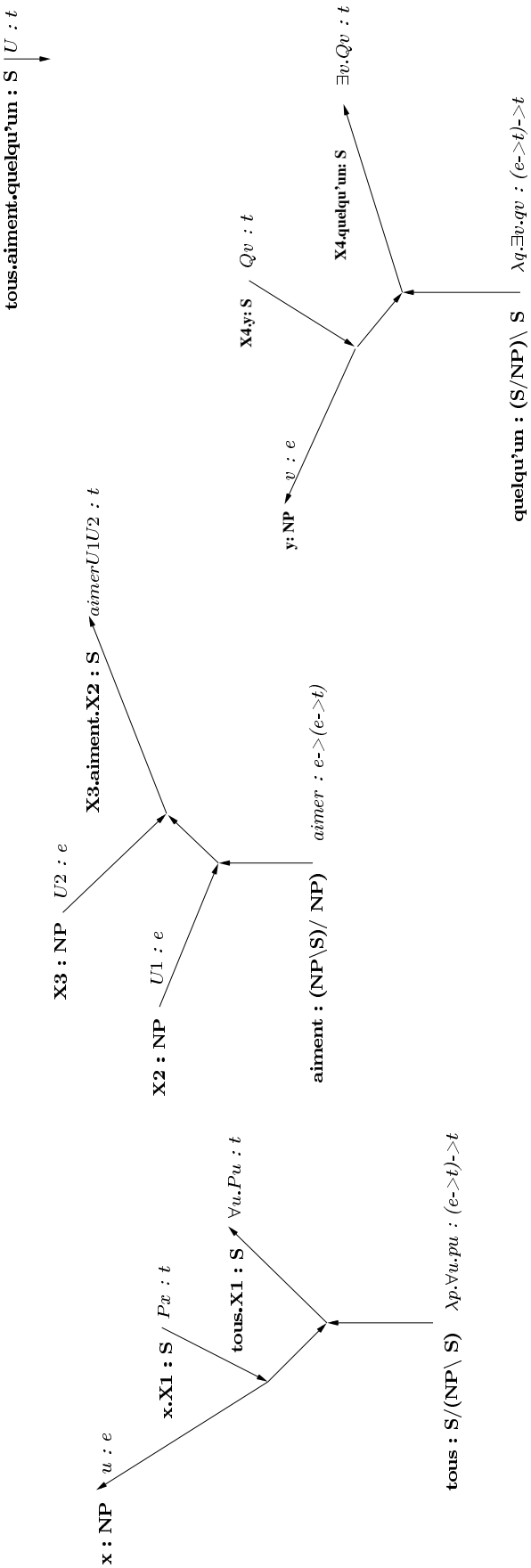


FIG. 4.2 – Structure de démonstration initiale correspondant à la phrase *« tous aiment quelqu'un »*

ce lexique. c'est pourquoi nous allons choisir la seconde option. Dans notre cas, nous effectuons une montée de type en remplaçant NP par un type plus spécifique. Nous attribuons à *tous* le type d'un syntagme nominal en début de phrase $S/(NP\backslash S)$ et à *quelqu'un* celui d'un syntagme nominal en fin de phrase $(S/NP)\backslash S)$. On voit tout le côté artificiel de cet exercice qui est dépendant de la position des quantificateurs dans la phrase : si *tous* est en fin de phrase, on devra lui attribuer un autre type et s'il est en position médiane, on se trouvera devant le même problème que pour l'extraction médiane des relatives. Effectuons néanmoins l'analyse avec les entrées lexicales telles qu'elles sont.

La structure de démonstration initiale correspondant à la phrase à analyser est donnée par la figure 4.2. Les étiquettes des feuilles des arbres ont été calculées à partir des étiquettes des racines par application des règles présentées à la figure 4.1. La démonstration proprement dite, qui consiste à poser des liens axiomes va être dirigée par les étiquettes phonologiques : un lien ne peut être posé qu'entre deux feuilles dont les étiquettes phonologiques sont unifiables. Dans notre exemple, nous avons deux possibilités de poser ces liens, qui correspondent aux deux lectures de la phrase. Nous choisirons celle où la portée de *tous* englobe celle de *quelqu'un*.

Parallèlement, la pose d'un lien axiome va entraîner l'unification des étiquettes sémantiques correspondantes, ce qui fait que la représentation sémantique de la phrase va se construire pas à pas jusqu'à ce que le réseau soit achevé. Cette représentation va alors se lire comme l'étiquette sémantique de la conclusion du réseau. La figure 4.3 nous montre ce réseau où nous ne faisons figurer que les étiquettes une fois qu'elles ont été unifiées[15]. On peut y lire que le sens de la phrase « *tous aiment quelqu'un* » est bien celui que l'on attendait, c'est-à-dire : $\forall u\ \exists v\ (aimer\ v\ u)$.

Ce petit exemple montre l'inconvénient qu'il y a à considérer la structure de dépendance sémantique comme une simple projection de la structure de dépendance syntaxique, ainsi que le font les grammaires catégorielles : pour rester cohérente avec la sémantique, la syntaxe doit être artificiellement compliquée.

4.2 Les descriptions de DAG polarisées

Dans les grammaires d'interaction synchrones, le niveau sémantique étant formalisé de façon similaire au niveau syntaxique, nous n'exposerons pas tous les détails de cette formalisation. Nous insisterons seulement sur les aspects originaux.

4.2.1 Les traits sémantiques polarisés

Nous disposons d'un ensemble fini $\mathcal{F}_s$ de noms de traits sémantiques. Dans la suite de l'exposé, nous considérerons $\mathcal{F}_s = \{type, val\}$. Chaque nom de trait f est associé à un ensemble $\mathcal{V}_f$ de valeurs atomiques. Par exemple, $\mathcal{V}_{type} = \{lex, logic, quant, ent\}$. Cela veut dire que le trait *type* représente un type

[15]Nous faisons figurer ces étiquettes sur les liens axiomes puisqu'après unification, les étiquettes des deux feuilles reliées par un lien axiome sont les mêmes.

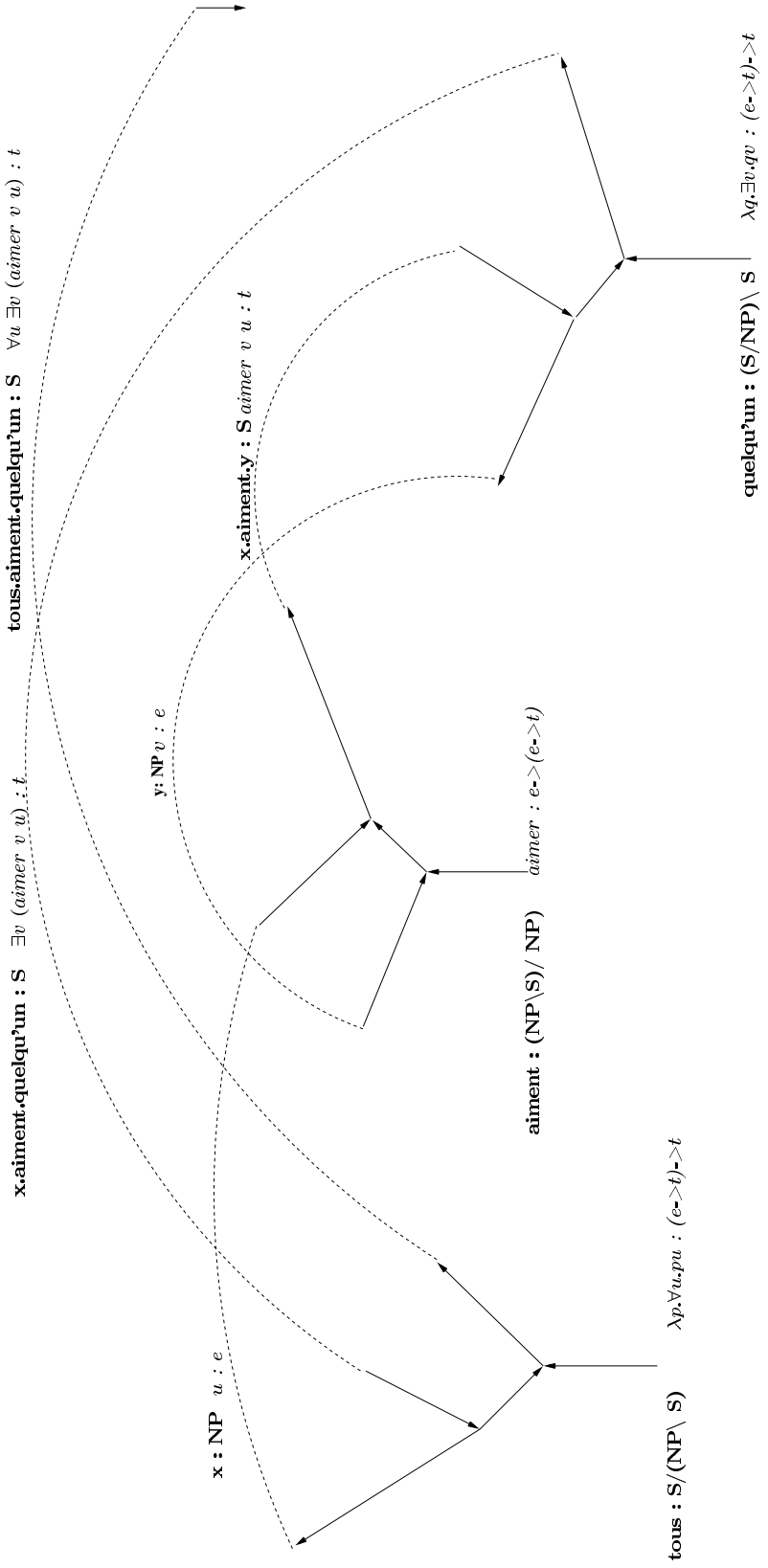


FIG. 4.3 – Réseau de démonstration correspondant à la phrase *« tous aiment quelqu'un »*

sémantique qui peut prendre les valeurs respectives *lex, logic, quant, ent*, qui sont des abréviations de *prédicats lexicaux*, *opérateurs logiques*, *quantificateurs* et *individus (ou entités)*. Le trait *val* représente la valeur sémantique de l'objet considéré. Voici quelques exemples d'éléments de $\mathcal{V}_{val}$: *voir, dormir, and, or, not, forall, exists, j, p*.

Comme pour les traits syntaxiques, on associe à chaque trait f le treillis distributif $\mathcal{D}_f$ de ses valeurs possibles qui sont des disjonctions finies de valeurs atomiques de $\mathcal{V}_f$. Pour les valeurs qui reviennent très souvent, nous utiliserons des abréviations. Par exemple, $pred = lex \cup logic \cup quant$.

On définit ensuite des structures de traits polarisés sémantiques effectives ou sur un environnement. Un trait polarisé est un triplet (nom de trait, polarité, variable) s'il est défini sur un environnement ou (nom de trait, polarité, valeur) s'il est effectif.

Par rapport à la syntaxe, se pose la question d'assouplir la notion de polarité pour permettre à un trait sémantique non neutre d'intervenir dans plus d'une neutralisation. Par exemple, dans la phrase *« Jean qui entre regarde Marie »*, le nœud sémantique représentant l'individu *Jean* doit être doublement capable de fournir un trait $type \rightarrow ent$, une fois comme agent de *entre* et une autre fois comme agent de *regarde*.

Une solution consiste à étendre les polarités à des entiers relatifs quelconques. L'addition des polarités serait donc celle des entiers relatifs et l'opération élémentaire de composition sémantique consisterait non pas strictement à neutraliser deux traits duaux mais à unifier deux traits de même nom et de polarités opposées, le résultat n'étant pas forcément un trait neutre. Dans notre exemple, le nœud sémantique représentant l'individu *Jean* serait porteur d'un trait $(type, +2, ent)$.

On peut aussi conserver le même genre de polarités au niveau sémantique qu'au niveau syntaxique mais dans ce cas, on doit considérer les traits de nom *type* et de valeur *ent* comme étant généralement neutres.

Nous ne trancherons pas définitivement entre les deux alternatives mais dans la suite nous nous placerons dans la première. Pour les polarités -1, 0, $+1$, nous conserverons les notations $\leftarrow$, $=$, $\rightarrow$.

4.2.2 Les descriptions de DAG

La syntaxe des descriptions de DAG est la même que celle des descriptions d'arbres à la différence près qu'il n'y a plus de relations de préséance entre nœuds. En effet, les nœuds sémantiques représentent soit des prédicats (type *pred*), soit des individus (type *ent*). Les seules relations qui ont du sens entre ces nœuds sont les relations prédicat-argument qui sont représentées par des relations de parenté et les relations de portée qui sont représentées par des relations de domination.

Pour figer la liste des différents arguments d'un même prédicat, on considère la relation $N > [N_1, \ldots, N_p]$ où les nœuds $N_1, \ldots, N_p$ représentent les arguments de N. A la différence de la syntaxe, ces arguments sont ordonnés de façon à distinguer leurs rôles dans le prédicat. Graphiquement, les arcs correspondants seront étiquetés par le rang de l'argument associé dans la liste. Au niveau syntaxique, il n'est pas nécessaire d'ordonner les constituants immédiats

d'un syntagme pour distinguer leurs fonctions syntaxiques. Ces fonctions sont indiquées par un trait *funct* attachés à ces constituants. Étant donné la structure d'arbre, ces constituants ont un seul père et il n'y a pas d'ambiguïté sur le syntagme dans lequel ils occupent cette fonction. On ne pourrait pas attacher aux nœuds sémantiques un trait *role* pour indiquer leur rôle thématique dans un prédicat car un même nœud peut être argument de plusieurs prédicats, ce qui est exprimé par la structure de DAG.

Par ailleurs, vu que dans un DAG, un nœud peut avoir plusieurs parents, il est utile d'introduire la relation $\{N_1, \ldots, N_p\} > N$ pour exprimer la contrainte que l'ensemble des parents du nœud N est $\{N_1, \ldots, N_p\}$.

Enfin, les relations de portée pourront être contraintes de la même façon que les relations de domination au niveau syntaxique avec des structures de traits neutres les étiquetant.

De la même façon qu'une description d'arbre est une manière économique et souple de représenter et de caractériser un ensemble d'arbres syntaxiques, une description de DAG présente le même avantage mais pour représenter un ensemble de DAG sémantiques. Cette notion de DAG sémantique n'est pas classique comme celle d'arbre syntaxique donc il faut la préciser un peu.

Dans une approche ambitieuse, un DAG sémantique peut être vu comme une représentation particulière d'une formule logique classique qu'on pourrait reconstituer en partant de la conjonction de tous les prédicats qui sont des racines du DAG. Chaque nœud du DAG représentant un prédicat pourrait être traduit sous forme d'un prédicat logique s'il est un prédicat lexical, sous forme d'un connecteur logique s'il en représente un ou sous forme d'une formule complexe s'il représente un quantificateur, cette formule complexe étant bien entendu fonction de la nature du quantificateur. La formule logique générale obtenue exprimerait alors la sémantique logique de l'énoncé correspondant.

Pour illustrer cette relation entre DAG sémantique et formule logique, considérons l'énoncé constitué des deux phrases :

« Un gâteau arrive. Tous les enfants en veulent un morceau. »

Cet énoncé a deux lectures selon la portée relative des quantificateurs *Tous les* et *un* de *un morceau*. Le morceau de gâteau que veulent les enfants peut être le même pour tous ou il peut être particulier à chaque enfant selon que la portée du second quantificateur englobe celle du premier ou le contraire. Considérons cette seconde lecture. Le sens de l'énoncé pourrait alors être représentée par le DAG de la figure 4.4. A partir de ce DAG il est facile de retrouver la formule logique correspondante si on sait qu'un prédicat de type *quant* (quantificateur) a 3 arguments : l'individu quantifié x, la restriction *R(x)* et la portée *P(x)*. Ensuite, l'interprétation logique dépend de la valeur du quantificateur : pour *exist*, c'est $\exists x(R(x) \wedge P(x))$ et pour *forall*, c'est $\forall x(R(x) \Rightarrow P(x))$. La formule logique représentée par le DAG de la figure 4.4 est alors la suivante :

$$\exists x(gateau(x) \wedge arriver(x) \wedge \forall y(enfant(y) \Rightarrow \exists z(morceau(z,u) \wedge vouloir(y,z))))$$

Pour avoir la représentation sémantique complète de notre énoncé, il reste encore à résoudre les anaphores, ce qui consiste à lier les variables libres de la formule obtenue. Ici, il y a une seule anaphore qui correspond à la variable libre u et elle se résout en substituant x à u. La formule logique finale sera donc :

$$\exists x(gateau(x) \wedge arriver(x) \wedge \forall y(enfant(y) \Rightarrow \exists z(morceau(z,x) \wedge vouloir(y,z))))$$

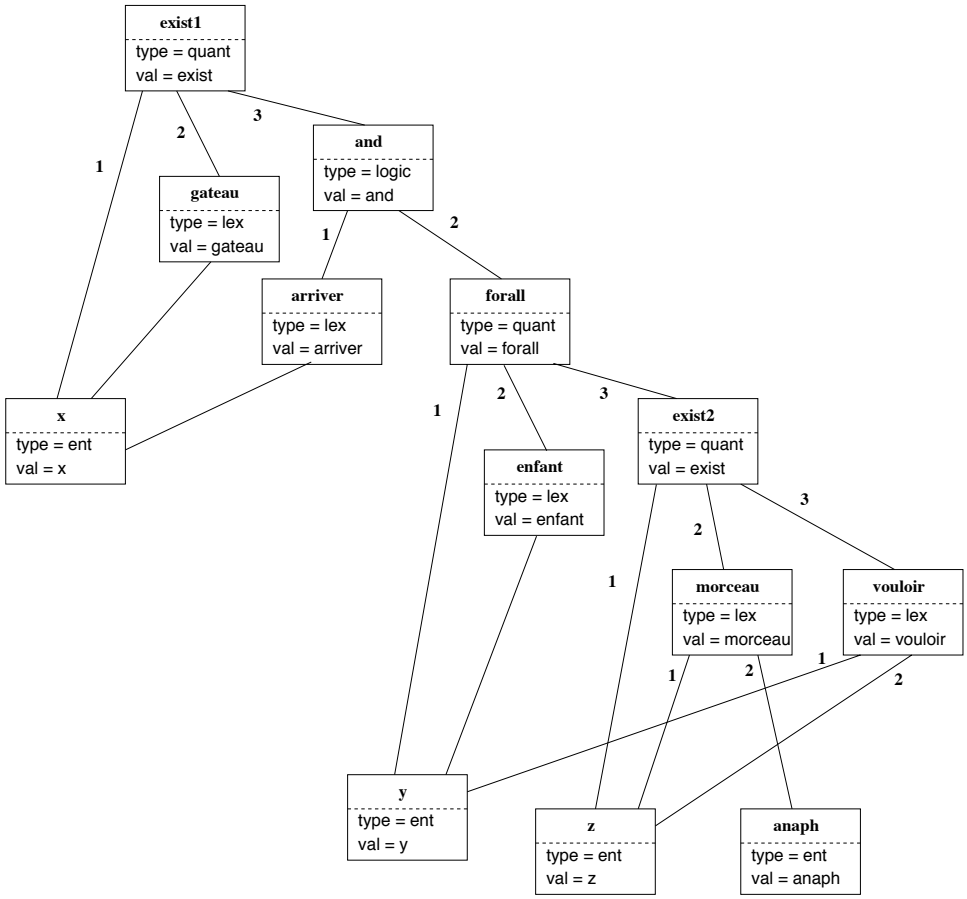


FIG. 4.4 – DAG représentant le sens de *Un gâteau arrive. Tous les enfants en veulent un morceau.*

Il semble très prétentieux de vouloir traiter complètement les problèmes de portée des quantificateurs à l'aide du formalisme des descriptions de DAG polarisés car les lois qui règlent les interactions entre portées sont très complexes. Déjà, comme le montre notre exemple, la notion de portée dépasse le strict cadre de la phrase. Ensuite, elle résulte souvent d'une interaction subtile entre syntaxe et sémantique. Considérons seulement la phrase simple *« Tous les hommes aiment une femme. »* qui a deux lectures. Si nous mettons cette phrase au passif, nous obtenons *« Une femme est aimée de tous les hommes. »* et cette phrase n'a plus qu'une seule lecture.

Pour traiter ce genre de problème, on peut, dans une première approche grossière, ignorer les contraintes de portée difficiles à formaliser. On obtient alors des représentations sémantiques sous-spécifiées dont certains modèles sont acceptables et d'autres pas. Il faut ensuite filtrer d'une façon ou d'une autre les modèles qui sont corrects. Une alternative consiste à exclure la représentation de la portée des quantificateurs de notre formalisme. La détermination de la portée de chaque quantificateur nécessitera alors un calcul indépendant comme la résolution des anaphores. Dans ce cas, les quantificateurs seront considérés comme des prédicats à deux arguments : l'individu quantifié et la restriction de quantification. Nous ne tranchons pas définitivement entre l'une ou l'autre des alternatives mais dans la suite de l'exposé nous choisirons, sauf exception, la seconde, c'est-à-dire de ne pas faire apparaître la portée des arguments. Cela permet d'alléger les présentations.

La figure 4.5 nous montre le DAG sémantique associé à l'énoncé *« Un gâteau arrive. Tous les enfants en veulent un morceau. »* où la portée des quantificateurs n'apparaît plus. Nous dirons que ce DAG est une représentation sémantique primitive de l'énoncé.

Formellement, un DAG sémantique est un DAG fini dont tous les nœuds sont étiquetés par des structures de traits de la forme $f{=}v$ où f est un élément de $\mathcal{F}_s$ et v un élément de $\mathcal{V}_f$. En outre, les fils d'un même nœud sont différenciés à l'aide d'une relation d'ordre total.

La relation entre description sémantique et DAG sémantique est établie par la notion de modèle qui est définie ainsi :

Définition 4.2.1. *Étant donné une description sémantique D, un modèle de cette description est un couple (G, I) où G est un DAG sémantique et I une fonction d'interprétation de l'ensemble $|D|$ des nœuds de D dans l'ensemble $|G|$ des nœuds de G qui vérifie les conditions suivantes :*

1. *I préserve toutes les relations de parenté et de domination de D ;*
2. *pour tout nœud N de D et pour tout nom de trait f présent dans la structure de traits associée à N, la structure de traits associée à $I(N)$ contient un trait $f = v$ tel que $v \leq v_f(N)$;*
3. *pour toute variable $< x >$ de l'environnement Γ associé à D et appartenant à $\mathcal{E}_f$, si les structures de traits associées à deux nœuds quelconques N et M contiennent respectivement les traits $(f, p, < x >)$ et $(f, q, < x >)$ alors $I(M)$ et $I(N)$ sont étiquetés par une structure de traits contenant le même trait $f = v$.*
4. *si D contient un prédicat $N > [N_1, \ldots, N_p]$, les fils de $I(N)$ sont les nœuds $I(N_1), \ldots, I(N_p)$ tous distincts et ordonnés de cette façon ;*

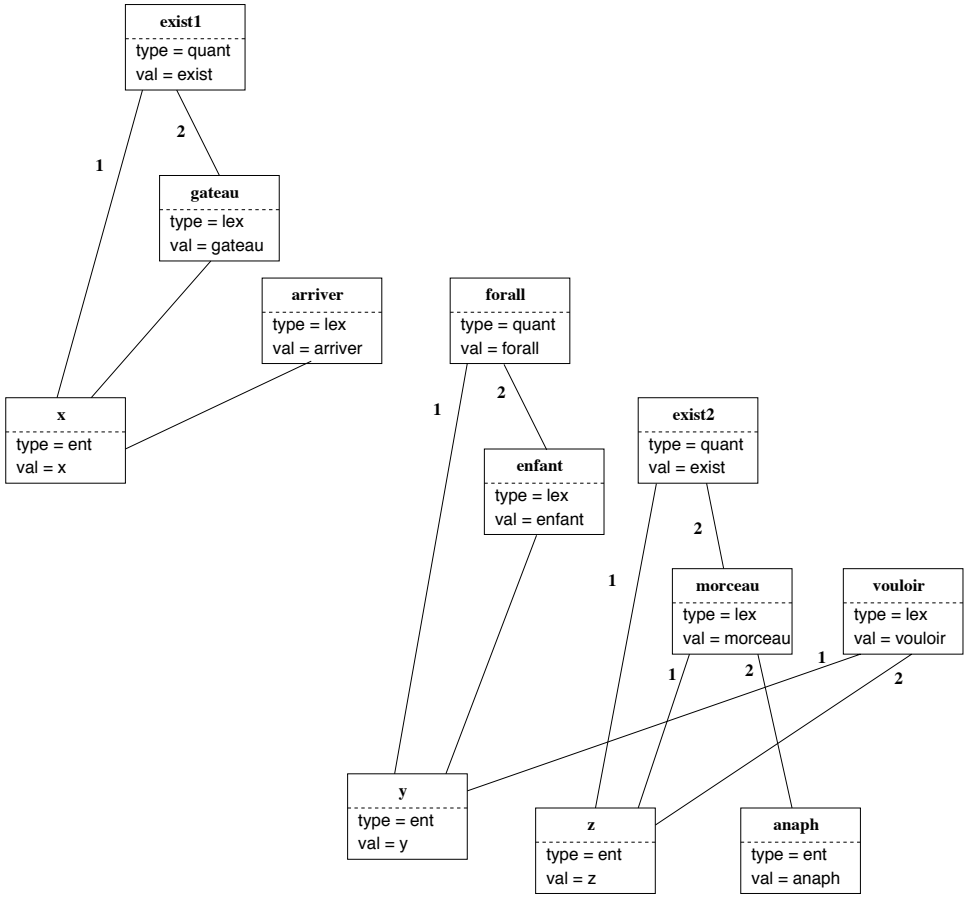


FIG. 4.5 – Représentation sémantique primitive de *Un gâteau arrive. Tous les enfants en veulent un morceau.*

5. *si D contient un prédicat $\{N_1, \ldots, N_p\} > N$, les pères de $I(N)$ sont les nœuds $I(N_1), \ldots, I(N_p)$ tous distincts ;*
6. *si D contient un prédicat $N_1 \overset{\star}{>}_S N_2$, alors tout nœud de G qui domine strictement $I(N_2)$ et qui est dominé par $I(N_1)$ au sens large doit être étiqueté par une structure de traits unifiable avec S.*

La notion de *modèles neutres et minimaux* de descriptions sémantiques se définit de la même façon qu'au niveau syntaxique donc nous ne reprendrons pas les définitions.

Illustrons la notion de modèle à l'aide d'un exemple. Considérons la phrase *« l'homme que Jean voit dort. »* et supposons qu'un lexique sémantique associe une description sémantique à chacun de ses mots d'où la spécification sémantique de la phrase donnée par la description de la figure 4.6. L'article défini *l'* est traité ici comme le quantificateur *« il existe un unique »*[16] et le pronom relatif *que* est utilisé de façon restrictive où *antec* est le prédicat exprimant le sens du nom commun qui le précède et *restrict2* est le prédicat exprimant le sens du verbe de la proposition relative. La représentation sémantique du point final comporte deux nœuds : *pred2* qui constitue la représentation sémantique de la proposition principale et *pred1* qui constitue la représentation sémantique de la phrase complète qui peut se réduire à celle de la proposition principale mais qui peut être plus complexe en raison de la présence de subordonnées par exemple. Bien entendu, cette description est hautement sous-spécifiée et c'est la syntaxe qui va introduire des contraintes sur les modèles neutres et minimaux pouvant la satisfaire. Parmi ces modèles, le DAG sémantique de la figure 4.7 fournit une représentation sémantique primitive de la phrase *« l'homme que Jean voit dort. »*. Pour en construire une achevée sous une forme logique, il nous faut connaître tout d'abord l'interprétation logique du quantificateur *unique*. Si x, *R(x)* et *P(x)* sont respectivement, la variable quantifiée, la restriction et la portée, l'interprétation est : $\exists x\ (R(x) \wedge (\forall y\ R(y) \Rightarrow y = x) \wedge P(x))$. Il faut aussi calculer la portée. Vu qu'il n'y a pas d'autre quantificateur dans la phrase, il n'y a pas d'ambiguïté et $P(x) = dormir(x)$, d'où la représentation sémantique logique de la phrase :

$\exists x(homme(x) \wedge voir(j, x) \wedge (\forall y(homme(y) \wedge voir(j, y) \Rightarrow x = y)) \wedge dormir(x))$

Une fois établie la notion de modèle, nous pouvons comme au niveau syntaxique comparer des descriptions sémantiques à l'aide de relations de *raffinement* et d'*équivalence*.

4.3 La synchronisation entre syntaxe et sémantique

Jusqu'à maintenant, nous avons défini les niveaux syntaxiques et sémantiques de façon indépendante. Bien évidemment, ces deux niveaux interagissent et, pour obtenir le formalisme complet des *grammaires d'interaction synchrones*, il est nécessaire de définir le mode de synchronisation des deux niveaux. Cela se fait très simplement par une fonction partielle qui projette chaque nœud

[16] Ce quantificateur est souvent noté en logique des prédicats $\exists!$ et la formule $\exists! x P(x)$ est alors une abréviation de $\exists x(P(x) \wedge (\forall y P(y) \Rightarrow y = x))$.

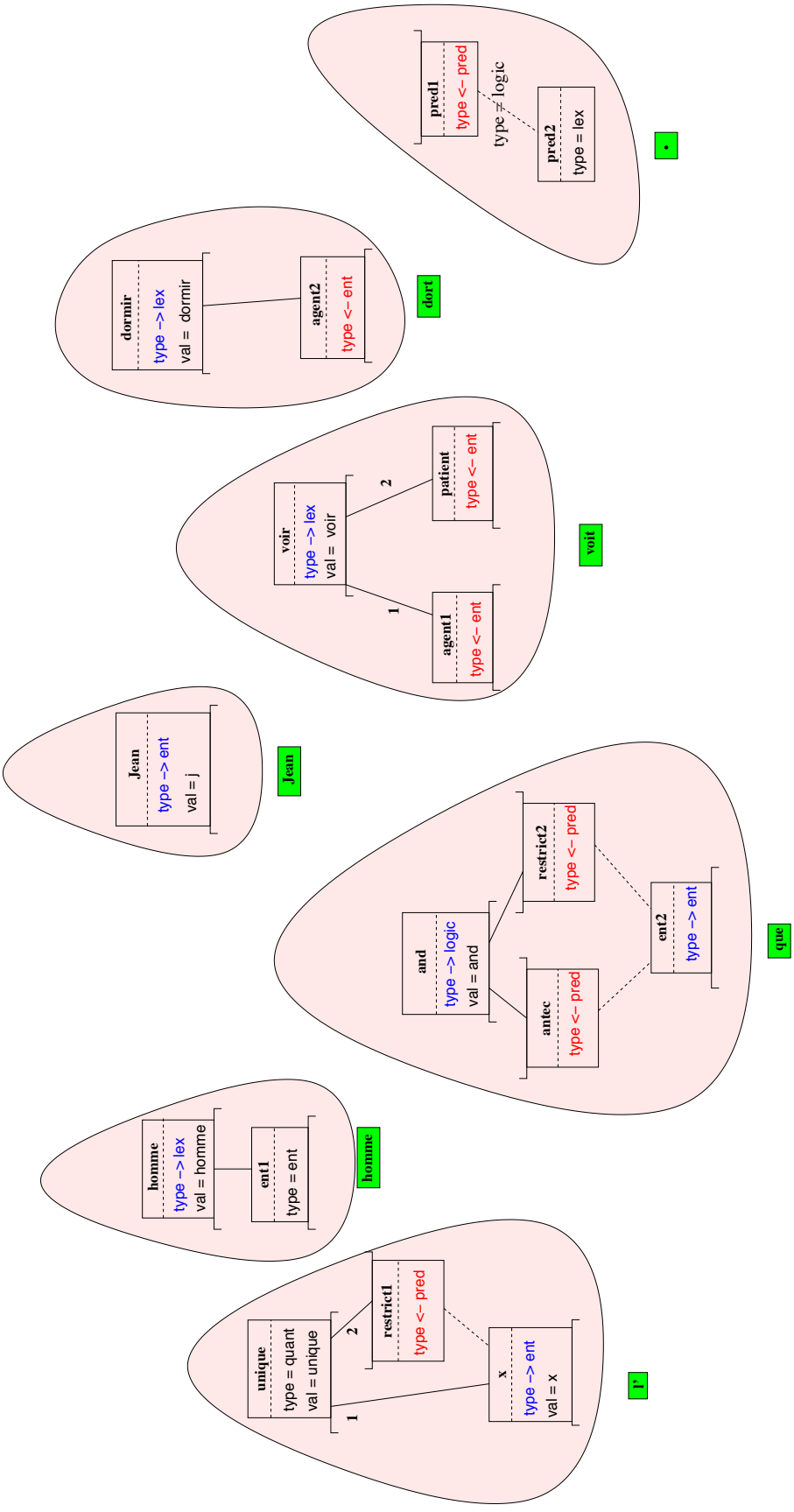


FIG. 4.6 – Description représentant la spécification sémantique de la phrase *« l'homme que Jean voit dort. »*

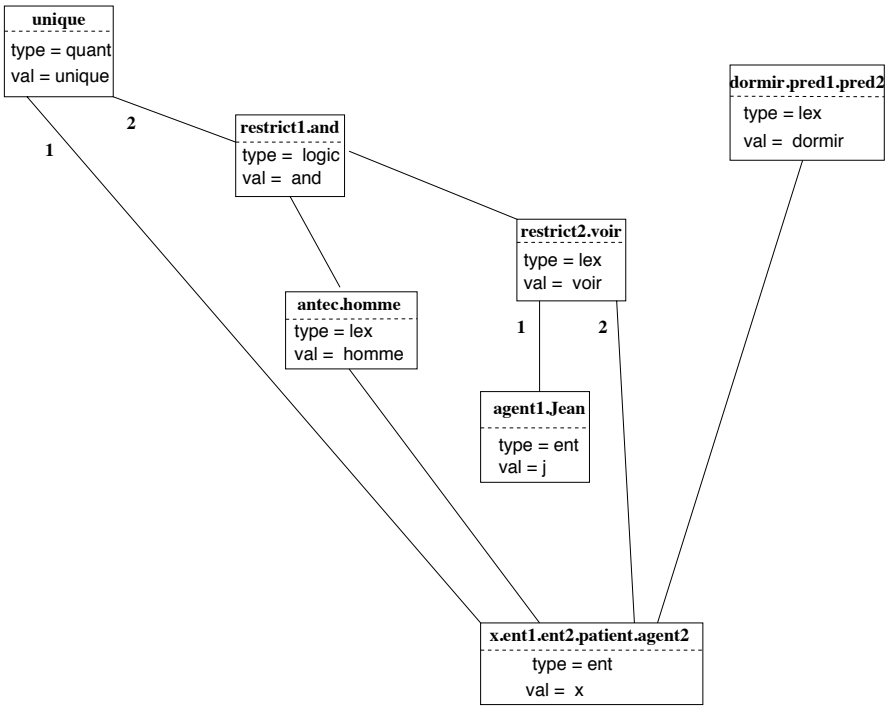


FIG. 4.7 – Représentation sémantique primitive de la phrase *« l'homme que Jean voit dort. »*

syntaxique sur au plus un nœud sémantique. Cela signifie que les nœuds syntaxiques soit ont une seule interprétation sémantique, soit ils n'en ont pas. Cela signifie aussi que les nœuds sémantiques interprètent de zéro à un nombre quelconque de nœuds syntaxiques. Tout ceci est valable tant au niveau des descriptions que des modèles.

Pour illustrer ceci, reprenons la phrase *« l'homme que Jean voit dort. »* et commençons par présenter la spécification syntaxico-sémantique de cette phrase qui pourrait être fournie par un lexique. Cette spécification est donnée par la figure 4.8. Les traits en pointillés représentent la fonction de synchronisation entre niveau syntaxique et niveau sémantique. On remarque que certains nœuds syntaxiques comme celui représentant le mot grammatical *que* n'ont pas d'interprétation sémantique et que des nœuds sémantiques comme les prédicats exprimant la quantification *unique* et *forall* n'interprètent aucun nœud syntaxique.

La figure 4.9 nous montre maintenant un modèle de cette description syntaxico-sémantique sous forme d'un arbre syntaxique couplé à un DAG sémantique. La fonction d'interprétation bien entendu doit préserver la correspondance entre nœuds syntaxiques et nœuds sémantiques.

Maintenant, si on envisage les choses de façon dynamique, on sait que la construction de la partie syntaxique du modèle se réalise par une suite de neutralisation de traits. Or, quand deux nœuds syntaxiques fusionnent pour réaliser une neutralisation, leurs projections sémantiques, si elles existent, fu-

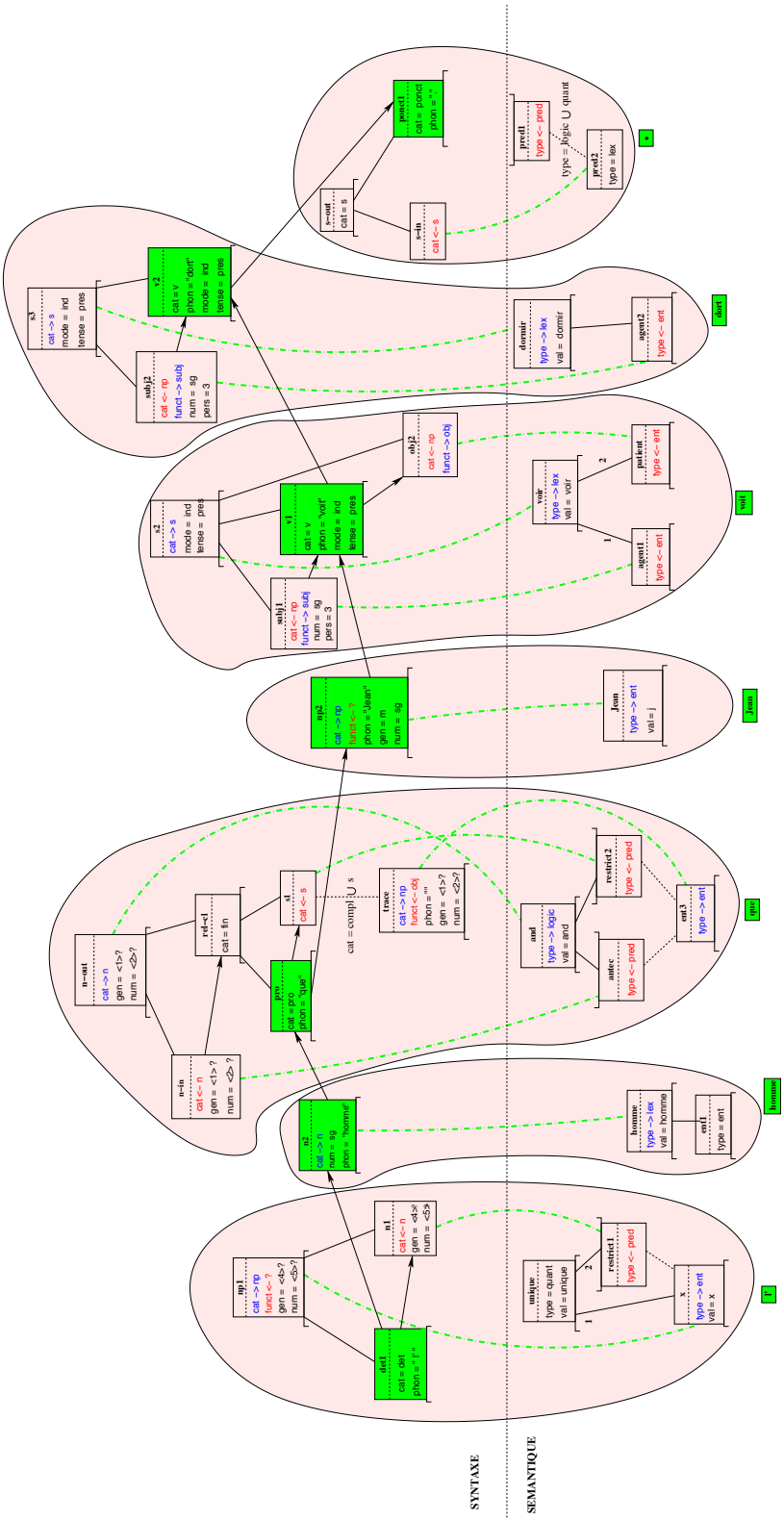


FIG. 4.8 – Spécification syntaxico-sémantique de la phrase *« l'homme que Jean voit dort. »*

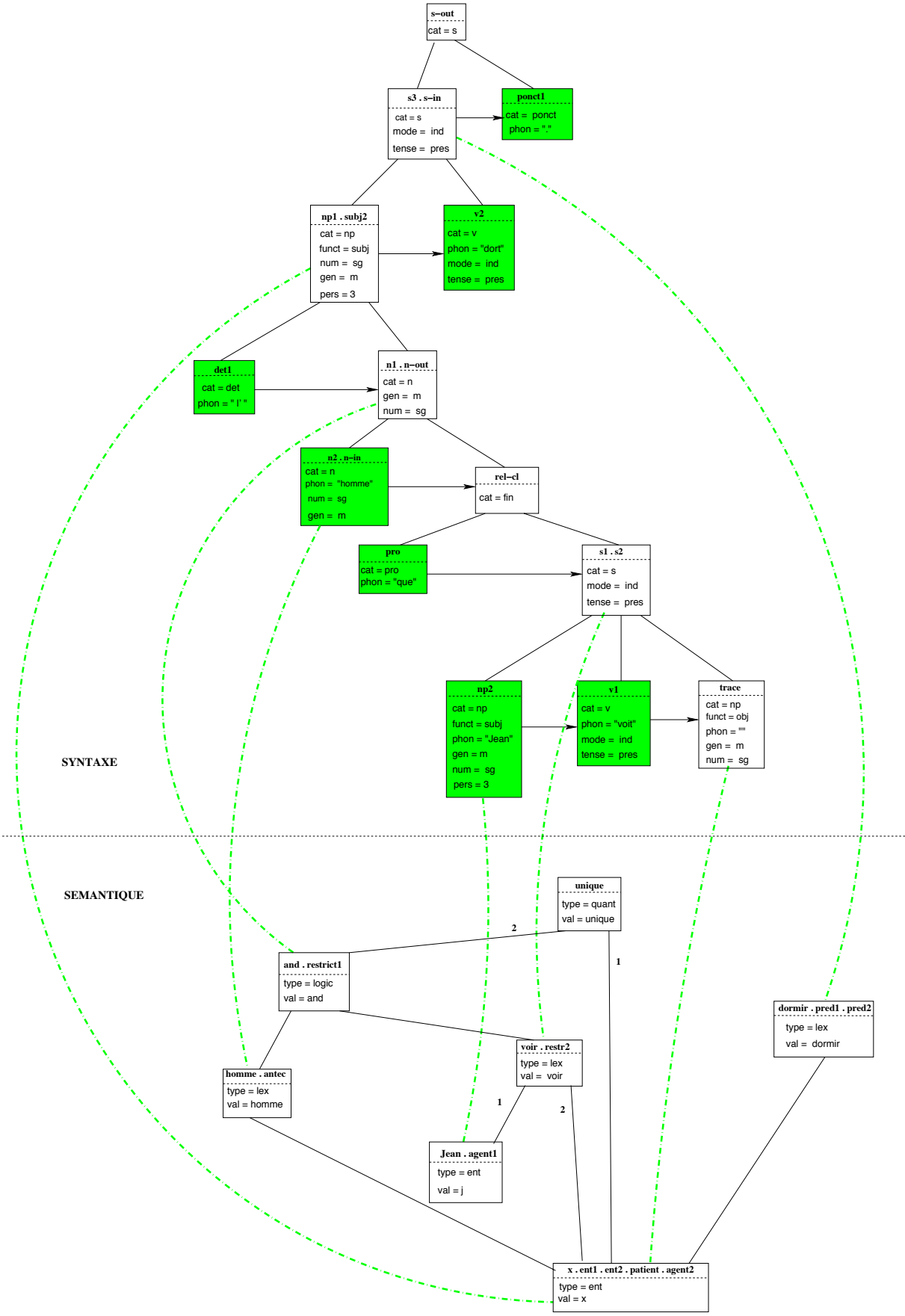


FIG. 4.9 – Représentation syntaxico-sémantique de la phrase *« l'homme que Jean voit dort. »*

sionnent aussi. Ainsi, la construction du modèle sémantique s'effectue de façon complètement automatique en parallèle du processus d'analyse syntaxique. Cela ne veut pas dire que cela se fasse de manière passive : une contrainte dans la description sémantique peut par exemple empêcher une fusion de deux nœuds syntaxiques et lorsque l'analyse syntaxique est achevée, cela ne veut pas dire pour autant que la construction du modèle sémantique soit aussi terminée. Il peut par exemple rester des traits sémantiques à neutraliser en fin d'analyse. Dans l'exemple que nous venons de présenter, à la fin de l'analyse syntaxique, il ne reste que deux traits sémantiques à neutraliser : les traits $type \rightarrow lex$ et $type \leftarrow pred$ des nœuds *dormir.pred2* et *pred1* et il n'y a qu'une façon de le faire en fusionnant ces nœuds.

Prenons un exemple plus intéressant de ce point de vue qui montre que la description sémantique obtenue en fin d'analyse syntaxique peut encore être ambiguë et donner lieu à plusieurs modèles neutres et minimaux. Il s'agit justement de la phrase classique *« tous aiment quelqu'un. »* qui nous a servi à mettre en lumière les limites des grammaires catégorielles. La figure 4.10 nous montre la spécification syntaxico-sémantique fournie par un lexique pour cette phrase. Contrairement à la ligne de conduite que nous nous étions fixés, nous avons représenté ici la portée des quantificateurs parce qu'elle va nous aider à établir la comparaison avec représentation à l'aide des grammaires catégorielles Déjà, nous n'avons pas eu besoin de changer le type de *tous* et *quelqu'un* pour rester en cohérence avec la sémantique. Ces deux mots ont la catégorie grammaticale *np*.

A la fin de l'analyse syntaxique, comme le montre la figure 4.11, lorsque tous les traits syntaxiques ont été neutralisés, nous obtenons un seul arbre syntaxique, à la différence des grammaires catégorielles où nous en obtenons deux. L'ambiguïté a été circonscrite au niveau sémantique. Le DAG sémantique y apparaît comme sous-spécifié sous forme d'une description. En procédant de la même façon qu'au niveau syntaxique, on peut construire les modèles neutres et minimaux de cette description par une suite de neutralisations de traits. Ici, nous obtenons deux modèles qui sont donnés par la figure 4.12 et qui correspondent aux deux lectures possibles de la phrase : celui du haut représente la formule logique $\forall x \exists y\ aimer(x, y)$ et celui du bas à $\exists y \forall x\ aimer(x, y)$.

Cette phrase illustre bien le fait que la représentation sémantique que l'on obtient en fin d'analyse syntaxique peut encore être sous-spécifiée et qu'il est nécessaire de neutraliser les traits sémantiques polarisés restants pour obtenir toutes les interprétations de la phrase. On peut s'arrêter là si on souhaite conserver une représentation factorisée de toutes les interprétations, surtout lorsqu'elles sont nombreuses.

L'exemple précédent ne montre pas le rôle actif de la sémantique par rapport à la syntaxe. Pour le mettre en évidence, considérons la phrase incorrecte *« * Jean pleut. »*. La figure 4.13 nous montre une spécification syntaxico-sémantique de cette phrase fournie par un lexique. L'analyse va échouer au niveau sémantique car le trait positif $type \rightarrow ent$ de l'entité correspondant à *Jean* ne pourra pas être neutralisé. Par contre, la phrase *« il pleut. »* sera correctement analysée car le pronom *il* explétif n'a pas de nœud sémantique qui lui correspond.

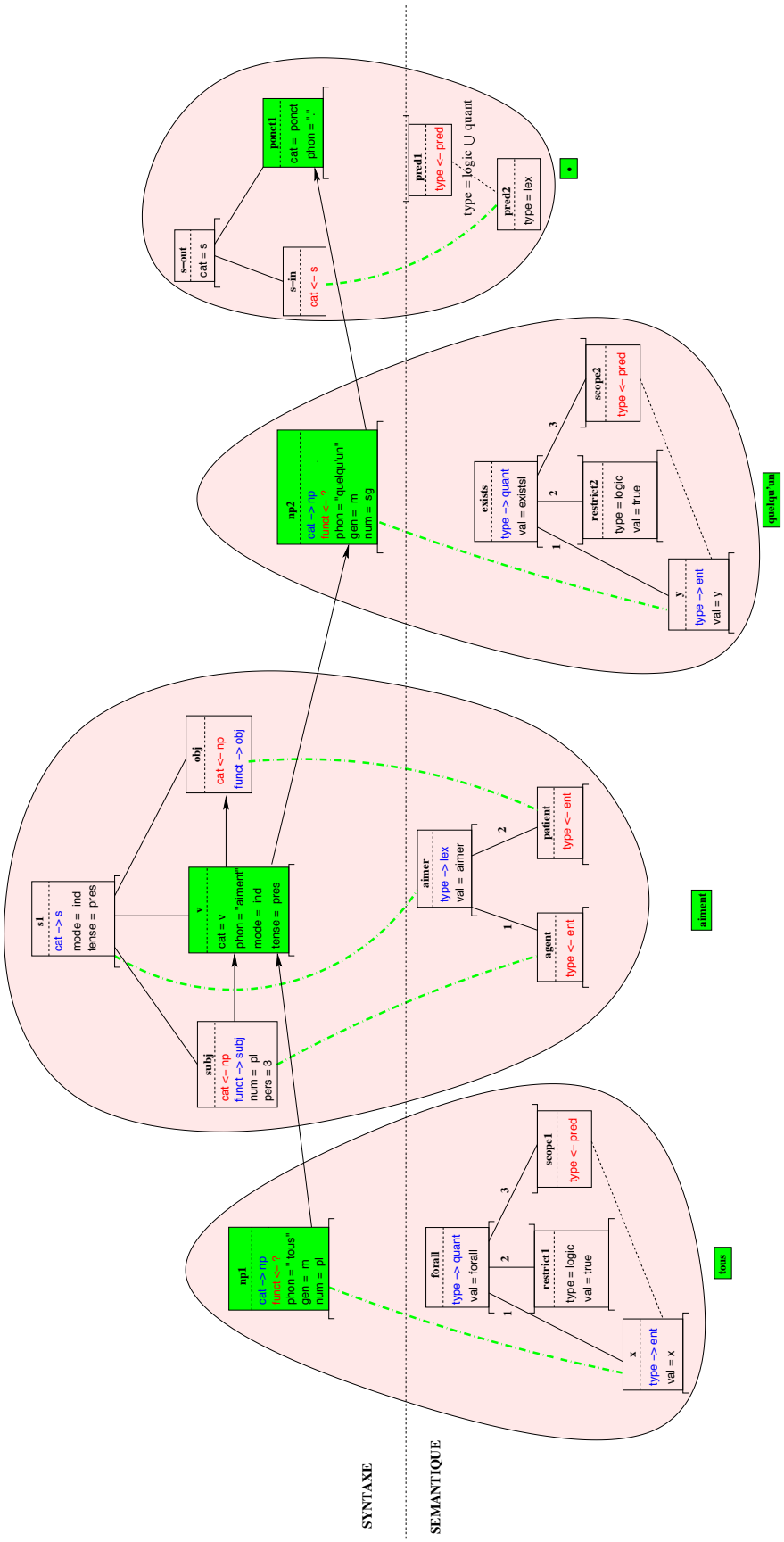


FIG. 4.10 – Spécification syntaxico-sémantique de la phrase *« tous aiment quelqu'un. »*

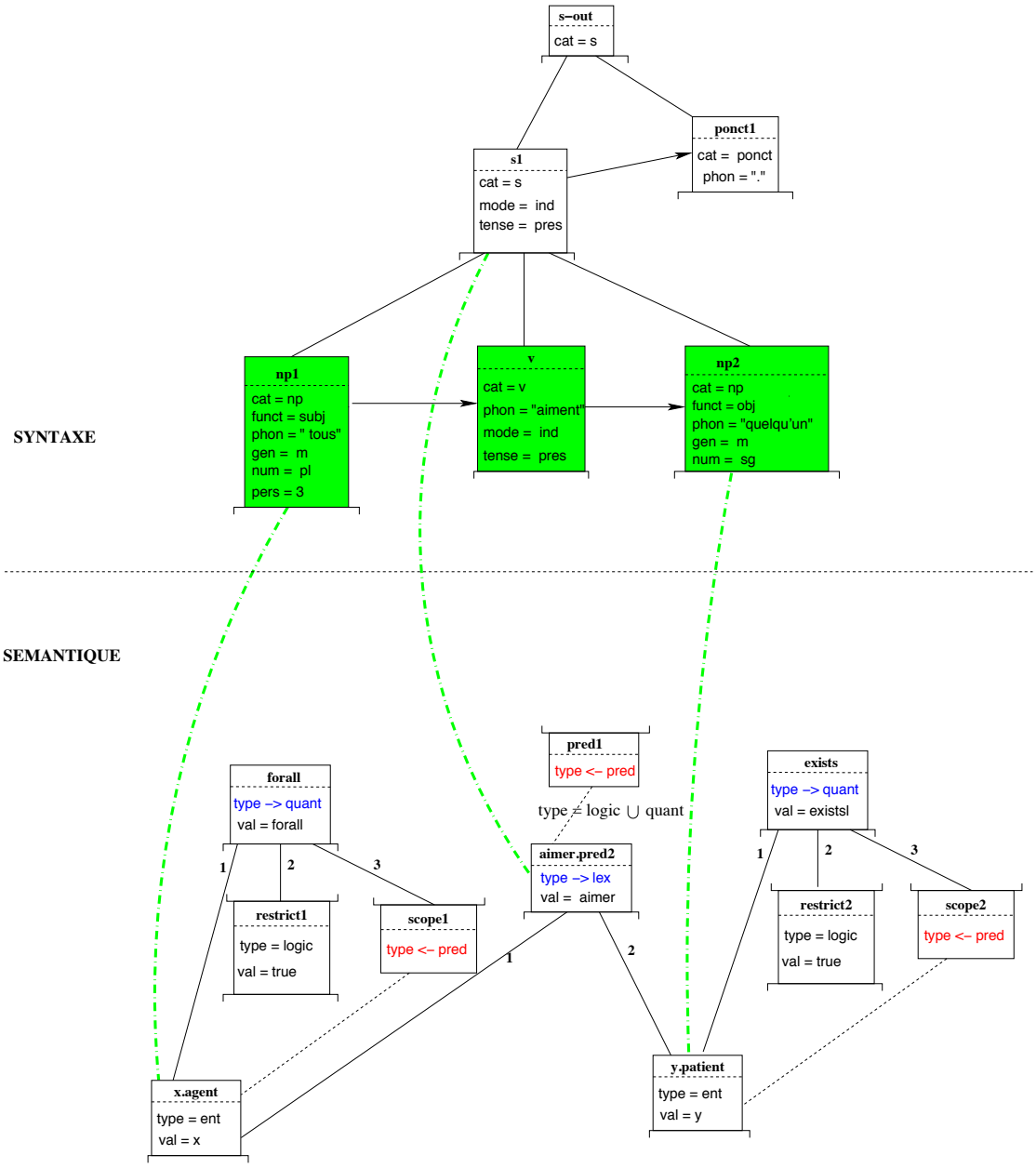


FIG. 4.11 – Description syntaxico-sémantique associée à *« tous aiment quelqu'un. »* à la fin de l'analyse syntaxique

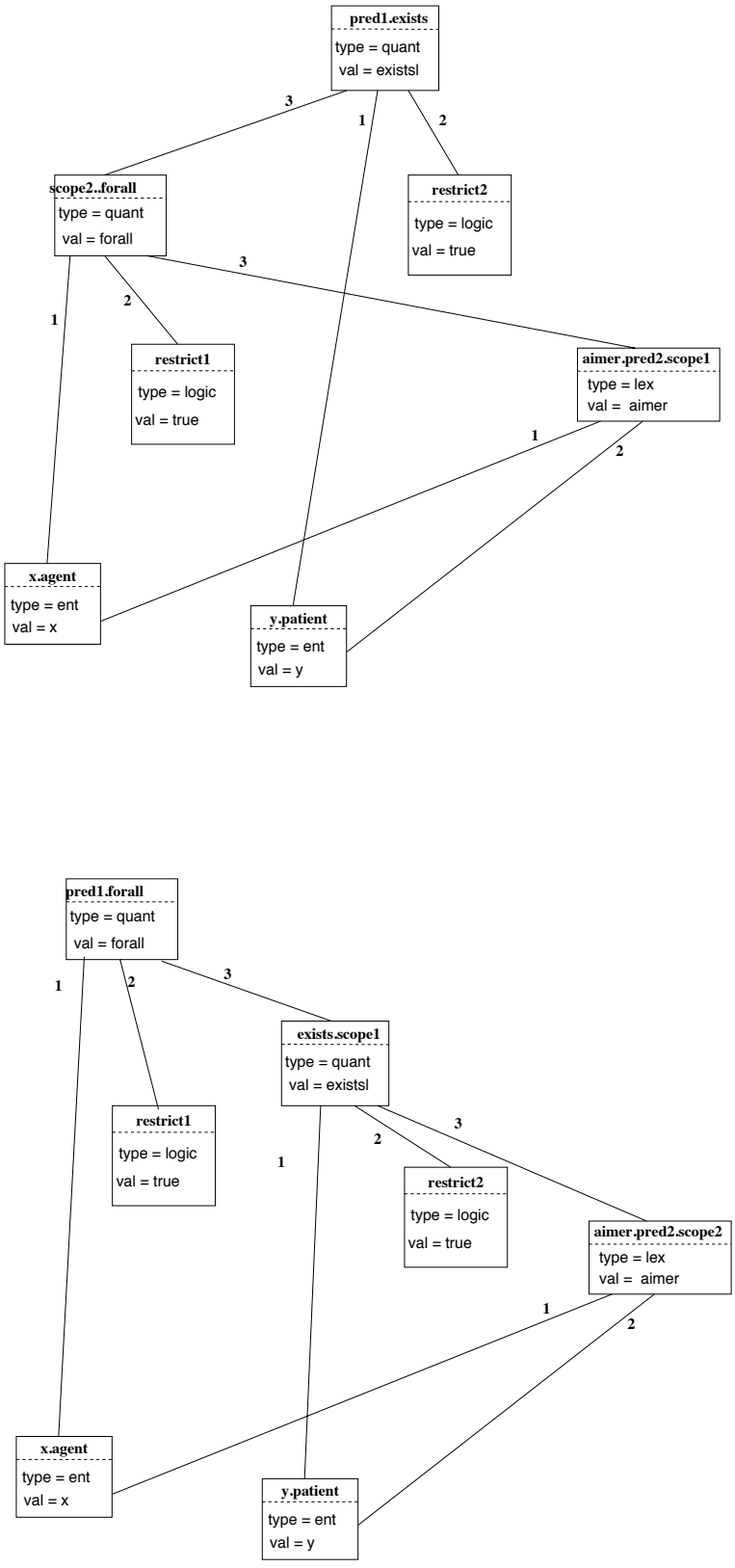


FIG. 4.12 – Représentations sémantiques possibles de *« tous aiment quelqu'un. »*

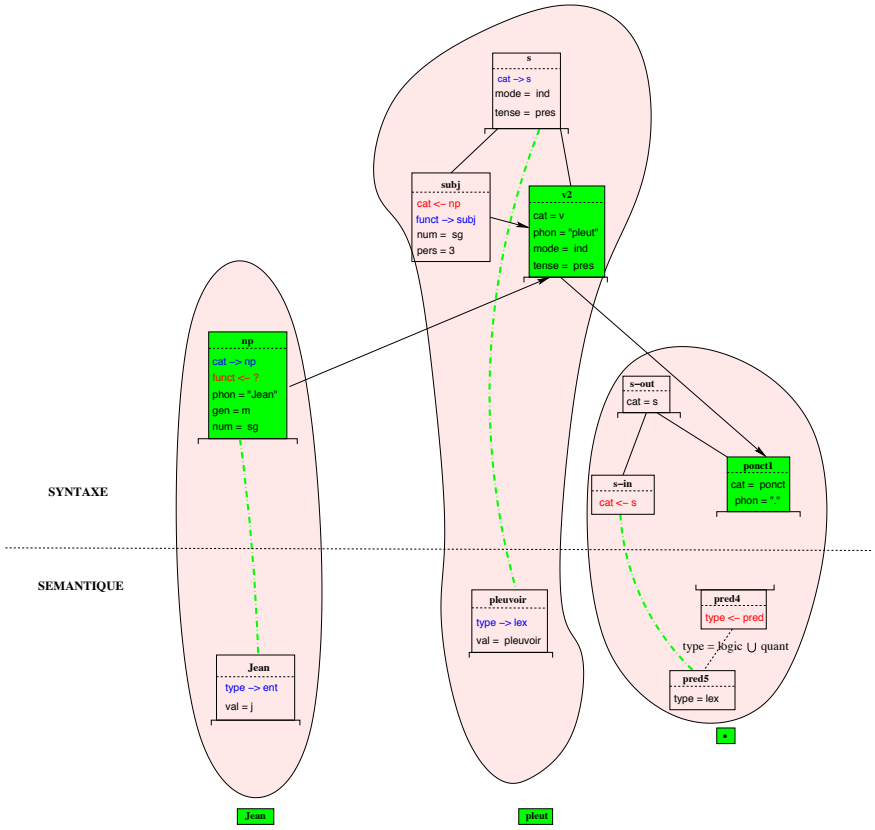


FIG. 4.13 – Spécification syntaxico-sémantique de la phrase incorrecte *« *Jean pleut.* »*

4.4 Exemples de modélisation de phénomènes syntaxico-sémantiques

4.4.1 Propositions relatives appositives et restrictives

Selon qu'elle sert ou pas à identifier le référent de son antécédent, une relative peut être restrictive ou appositive. Dans la section précédente, nous avons rencontré une relative dans la phrase *« l'homme que Jean voit dort. »* et nous l'avons interprété de façon restrictive. En conséquence, au niveau syntaxique, nous avons modélisé la relative *« que Jean voit »* comme un modifieur du nom commun *homme* et au niveau sémantique, nous l'avons interprété comme venant s'ajouter à *homme* pour venir déterminer la restriction du quantificateur *unique* associé à *l'*.

Les grammaires catégorielles offrent deux constructions syntaxiques possibles pour une relative : comme modifieur de l'antécédent si elles sont appositives et comme modifieur du nom commun qu'il contient si elles sont restrictives. Cette dualité de construction syntaxique est discutable et est contredite par des exemples tels que *« celui que Jean voit dort »* qui possède toujours deux interprétations mais un seul arbre syntaxique. Le fait que l'on ne puisse pas modifier par une relative un nom commun qui n'est pas précédé d'un dé-

terminant (on ne peut pas dire « * *Jean travaille avec habileté qu'ont les gens d'expérience* ») va dans le même sens : il semble raisonnable de considérer dans tous les cas une relative comme modifiant le syntagme nominal qui précède mais avec deux interprétations possibles. La souplesse des grammaires d'interaction va nous permettre d'avoir une seule description syntaxico-sémantique pour exprimer ces deux interprétations[17].

Reprenons la phrase *« l'homme que Jean voit dort. »* et considérons une nouvelle spécification syntaxico-sémantique de celle-ci fournie par un lexique. Elle nous est donnée par la figure 4.14. Par rapport à celle présentée sur la figure 4.8, deux entrées lexicales ont été modifiées : celle de *l'* où, au niveau sémantique, la restriction du quantificateur ne se confond plus avec le prédicat correspondant au nom commun qui est déterminé par *l'* mais elle le domine seulement et l'entrée de *que* où la relative devient modifieur de groupe nominal au lieu de nom commun. A la fin de l'analyse syntaxique de la phrase, nous obtenons la description syntaxico-sémantique de la figure 4.15 où l'arbre syntaxique est complètement spécifié mais pas le DAG sémantique. Si on poursuit le calcul en neutralisant les traits sémantiques polarisés restants, on obtient deux DAG sémantiques, tels que le montre la figure 4.16. Celui du haut correspond à l'interprétation restrictive de la proposition relative : si nous désignons par x la référence de *l'homme*, le prédicat *voir(j,x)* vient restreindre le prédicat *homme(x)* pour former la restriction du quantificateur *unique* qui est alors $R_{unique}(x) = homme(x) \wedge voir(j,x)$. Le calcul de la portée $P_{unique}(x)$ est trivial puisque cette portée doit comprendre tous les prédicats dont x est un argument et qui ne sont pas dans $R_{unique}(x)$. On a donc : $P_{unique}(x) = dormir(x)$. Enfin, si on sait que le prédicat *unique* code la formule logique $\exists x R_{unique}(x) \wedge P_{unique}(x) \wedge \forall y(R_{unique}(y) \Rightarrow y = x)$, on obtient la formule logique qui constitue une interprétation possible de la phrase :

$$\exists x\ (homme(x) \wedge voir(j,x) \wedge dormir(x) \wedge \forall y\ ((homme(y) \wedge voir(j,y)) \Rightarrow y = x))$$

Le DAG sémantique du bas correspond à l'interprétation appositive de la proposition relative. La restriction du quantificateur *unique* est : $R_{unique}(x) = homme(x)$. Sa portée se calcule sans difficultés : $P_{unique}(x) = dormir(x) \wedge voir(j,x)$. En conséquence, l'autre interprétation possible de la phrase est représentée par la formule logique :

$$\exists x\ (homme(x) \wedge voir(j,x) \wedge dormir(x) \wedge \forall y\ (homme(y) \Rightarrow y = x))$$

4.4.2 Verbes à montée du sujet et verbes à contrôle du sujet

Les verbes à montée du sujet et les verbes à contrôle du sujet ont en commun de prendre comme complément une infinitive et leur sujet se confond avec celui de l'infinitive. La différence essentielle se fait au niveau sémantique. Considérons le verbe à contrôle *vouloir* et le verbe à montée *sembler* :

- le prédicat interprétant *vouloir* prend deux arguments, son agent qui est un individu x et son objet, un prédicat p représentant le sens du verbe complément ; en plus, x est un argument de p car il est la référence du sujet commun des deux verbes ;

[17] A ce sujet, il serait d'utiliser la méthode proposée par A. Abeillé dans [Abe02] pour déterminer le découpage en syntagmes dans ce cas précis.

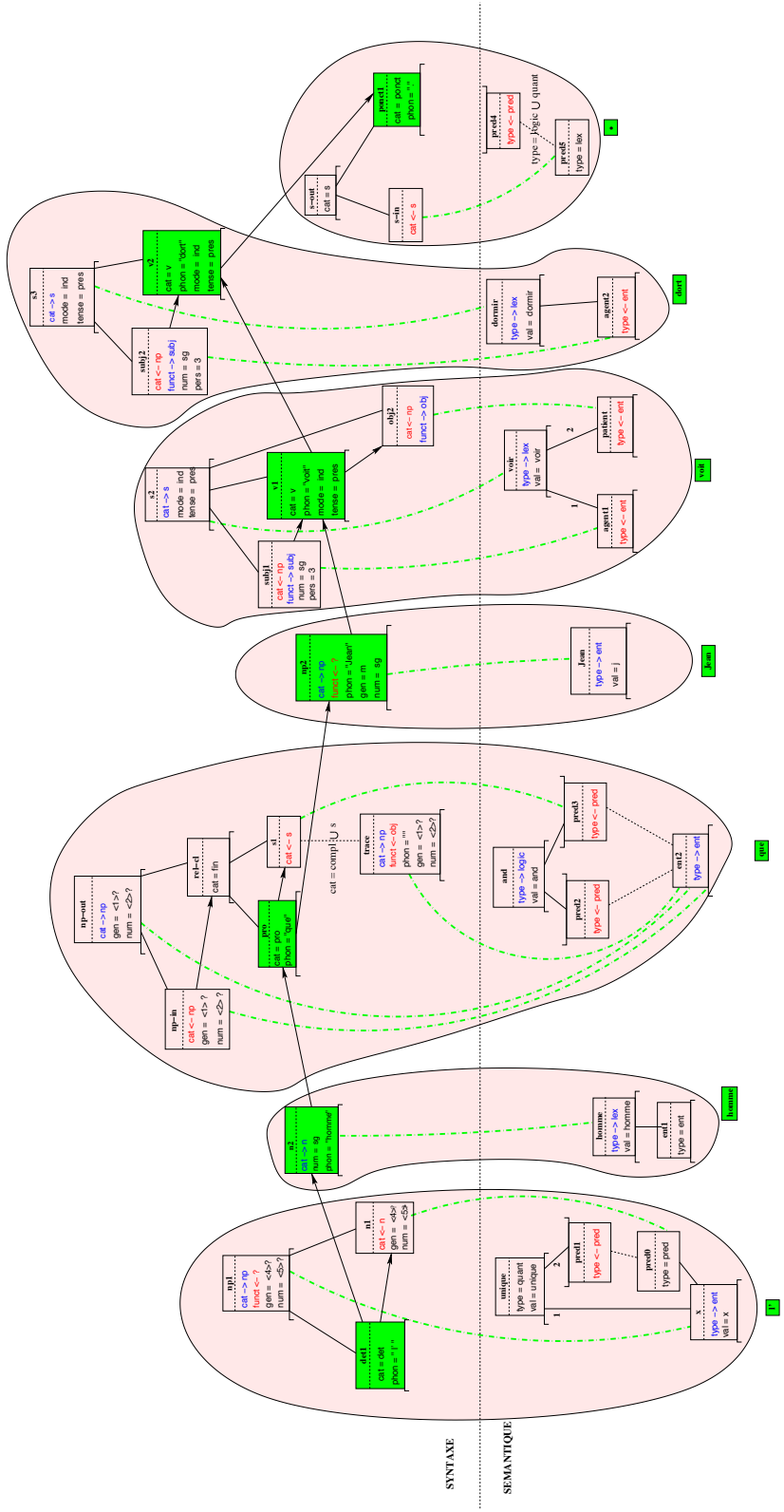


FIG. 4.14 – Nouvelle spécification syntaxico-sémantique de la phrase *« l'homme que Jean voit dort. »*

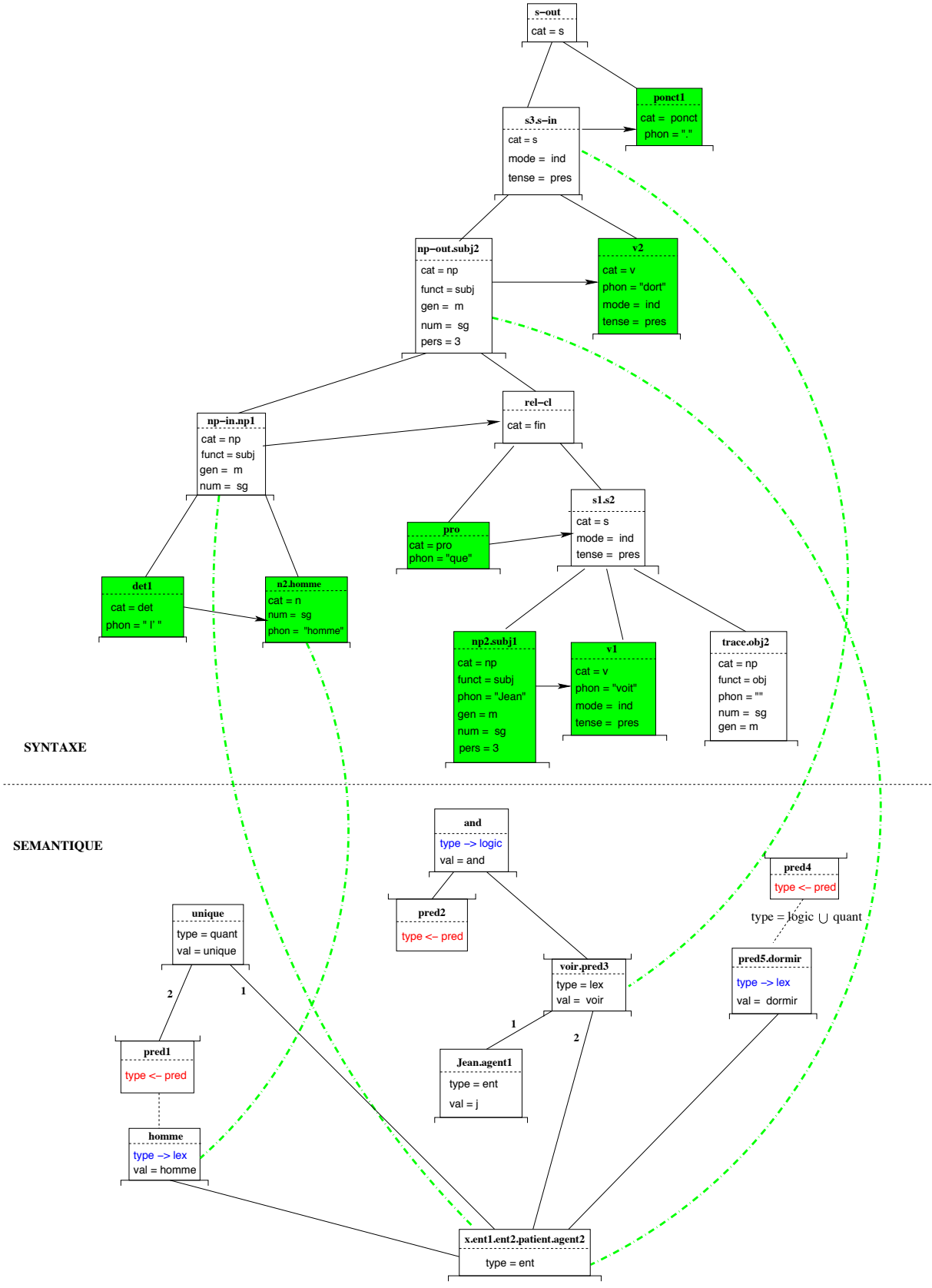
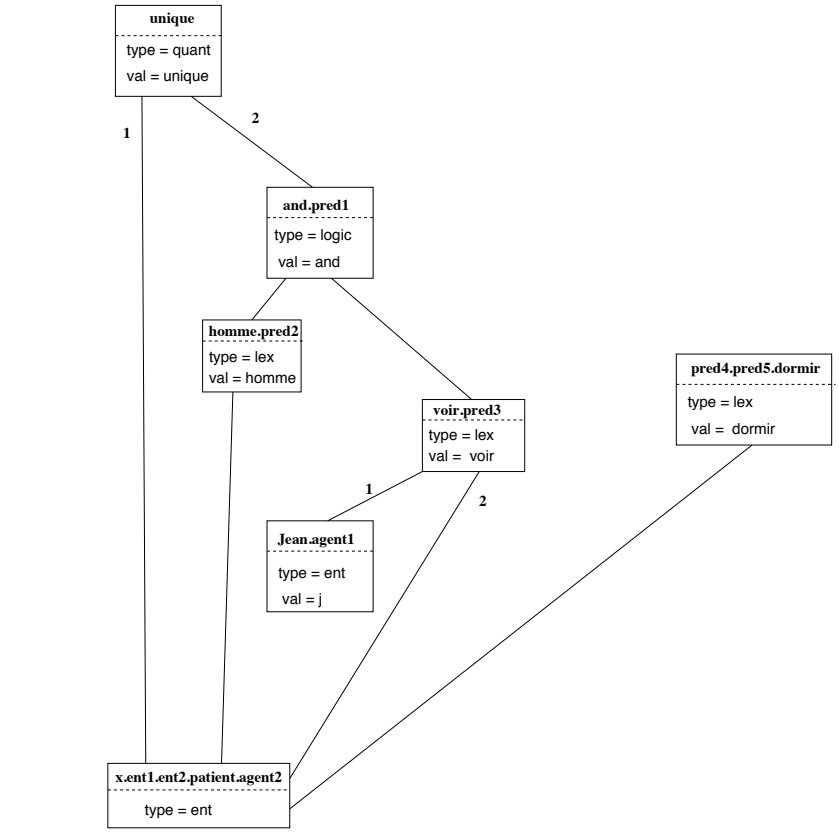


FIG. 4.15 – Description syntaxico-sémantique résultant de l'analyse syntaxique de la phrase *« l'homme que Jean voit dort. »*



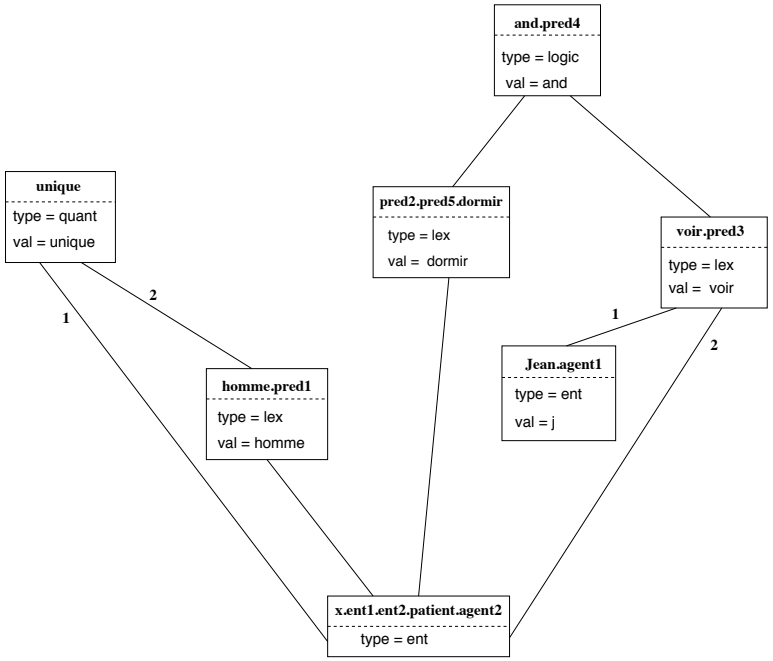


FIG. 4.16 – Représentations sémantiques primitives possibles de la phrase *« l'homme que Jean voit dort. »*

– le prédicat interprétant *sembler* prend un seul argument qui est le prédicat interprétant l'infinitive complément.

Au niveau syntaxique, ils ont un comportement très voisins même si on peut noter des différences mineures comme le signale A. Abeillé [Abe02]. Un certain nombre de phénomènes les rapprochent des auxiliaires de temps et nous empêche de les considérer comme des verbes ordinaires prenant une infinitive comme complément. Soit par exemple les phrases suivantes :

(a) l'homme que veut présenter Jean à Marie dort.
(b) l'homme que semble présenter Jean à Marie dort.

Si on considère que *veut* et *semble* prennent comme complément une infinitive, celle-ci *présenter à Marie* est interrompue par le sujet *Jean*, ce qui va à l'encontre de la continuité des syntagmes. Des raisons opposées pourraient nous inciter à les traiter comme des verbes ordinaires avec un complément qui est une infinitive partageant son sujet avec le leur. C'est par exemple l'absence de montée des clitiques portant sur les infinitives. On dit *« Jean veut la voir »* et non pas *« * Jean la veut voir »*.

En TAG, A. Abeillé [Abe02] représente les entrées lexicales des deux types de verbes par des arbres auxiliaires. Pour respecter le principe de co-occurrence prédicat-arguments, les arbres des verbes à montées viennent s'adjoindre sur des nœuds de type *verbe* tandis que les arbres des verbes à contrôle viennent s'adjoindre sur les nœuds de type *proposition*. Pour ces derniers, cela présente une difficulté qui consiste à réaliser l'identification des deux sujets.

Nous n'avons pas le même problème avec les grammaires d'interaction où l'on utilise le mécanisme de superposition d'arbres. Les deux types de verbe ont exactement la même représentation syntaxique qui se rapproche de celle d'un auxiliaire de temps. Ils ne diffèrent qu'au niveau sémantique et de l'interface syntaxe-sémantique.

La figure 4.17 donne une représentation syntaxico-sémantique des verbes *semble* et *veut* lorsqu'ils prennent un complément à l'infinitif. Leur niveaux syntaxiques sont identiques ; seuls diffèrent les niveaux sémantiques qui expriment ce qui vient d'être décrit.

Pour comprendre le fonctionnement de ces descriptions, considérons deux verbes à l'infinitif, *dormir* et *pleuvoir*, le premier étant personnel et le second impersonnel. La figure 4.18 donne une représentation syntaxico-sémantique de ces verbes. Nous avons modifié légèrement l'interface syntaxe-sémantique en attachant le prédicat représentant le sens du verbe à l'ancre verbale et non plus à la phrase car ce prédicat peut être modifié par un adverbe ou une négation. Avec les spécifications syntaxico-sémantiques fournies par les figures 4.17 et 4.18, nous pouvons analyser les phrases correctes et incorrectes suivantes :

Jean veut dormir.
Jean semble dormir.
Il semble pleuvoir.
** Jean semble pleuvoir.*
** Il veut pleuvoir.*
Jean semble vouloir dormir.
**Jean veut sembler dormir.*

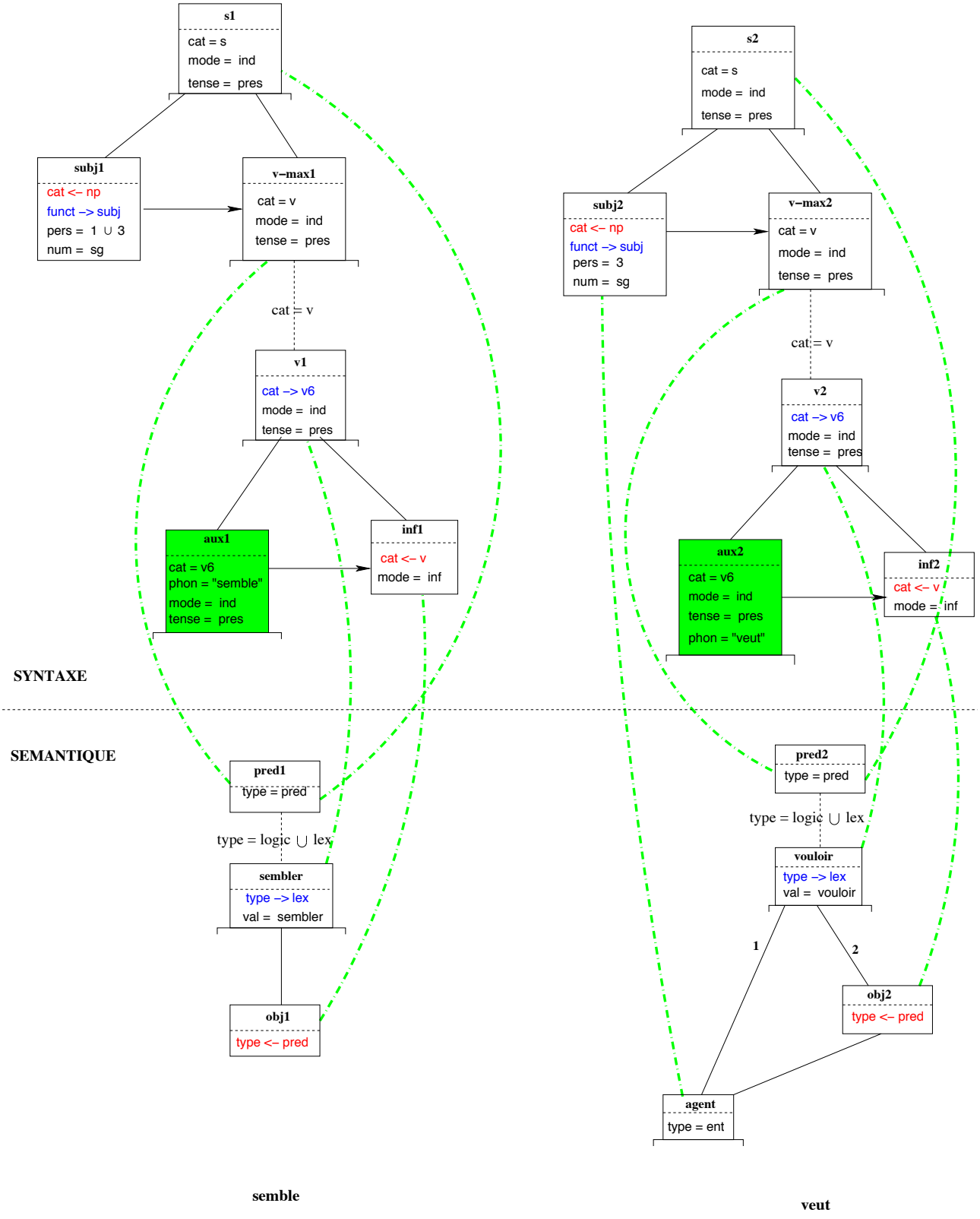


FIG. 4.17 – représentation syntaxico-sémantique des verbes *semble* et *veut*

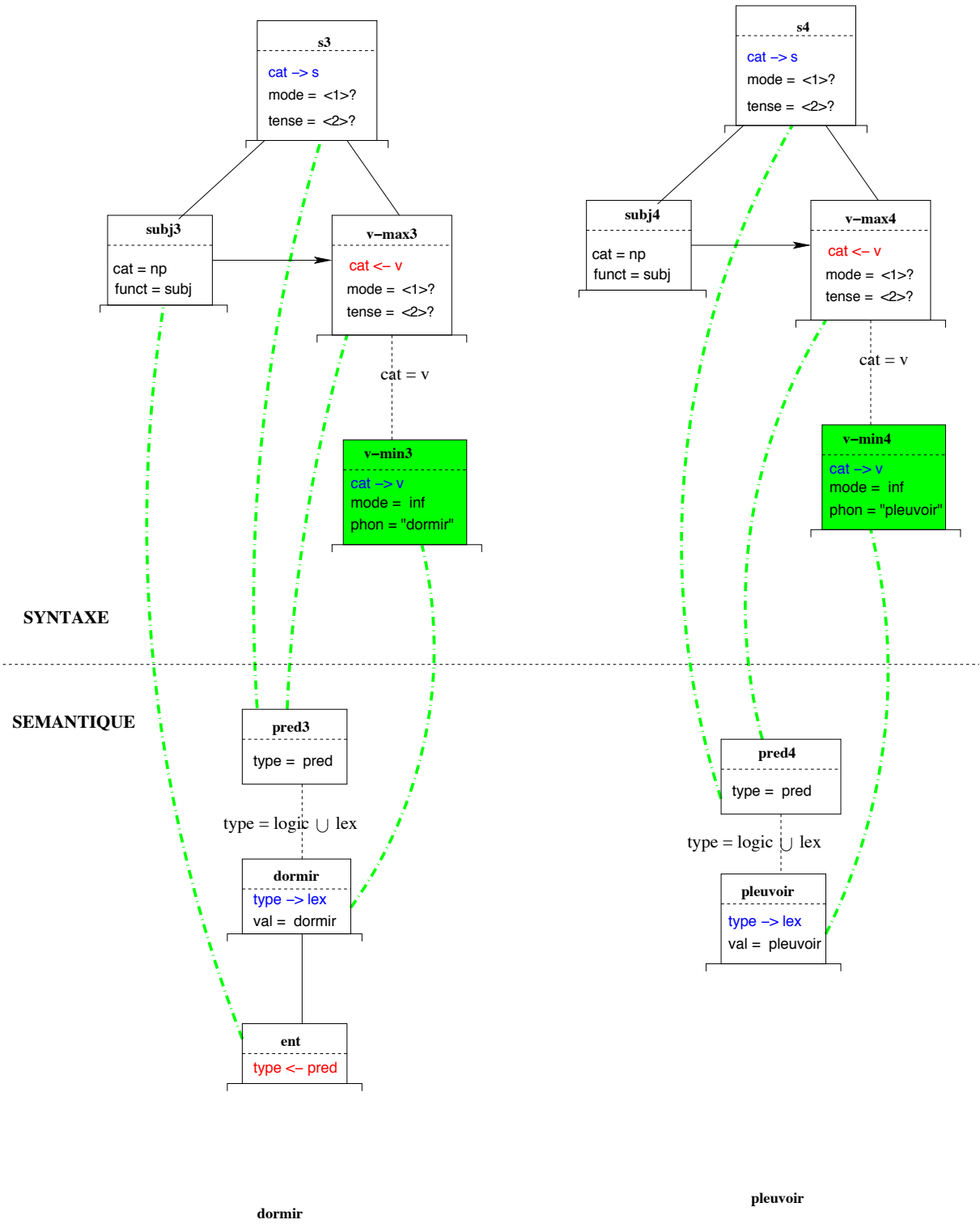


FIG. 4.18 – représentation syntaxico-sémantique des verbes *dormir* et *pleuvoir*

Pour les deux dernières phrases, il faut modifier légèrement les descriptions associées à *veut* et *semble* pour avoir celles qui correspondent aux infinitifs *vouloir* et *sembler*.

On peut remarquer par ailleurs que les représentations syntaxiques de *veut* et *semble* sont compatibles avec notre représentation des clitiques telle qu'elle a été décrite à la section 2.3 et permet d'analyser correctement les phrases :

Jean semble l'avoir compris.
** Jean le semble avoir compris.*
Jean veut le lui présenter.
** Jean le lui veut présenter.*

4.4.3 Cas difficiles d'infinitives compléments d'adjectifs

Certains adjectifs comme *facile* admettent comme complément une infinitive comportant un objet direct qui représente en même temps l'entité qui est qualifiée par l'adjectif. Ce phénomène est désigné en anglais par le terme de *« tough movement »*. Ainsi dans l'expression *« un livre facile à lire »*, *« un livre »* est modifié par le syntagme adjectival *« facile à lire »* mais en même temps il est l'objet direct de *lire*. On peut faire l'analogie avec la relative *« un livre qu'on lit facilement »* et on retrouve la double lecture restrictive et appositive. C'est pourquoi nous proposons à la figure 4.19 une formalisation analogue comme modifieur de syntagme nominal au niveau syntaxique et avec une conjonction de prédicats au niveau sémantique et une sous-spécification qui autorise les deux lectures.

L'objet de l'infinitive complément de *facile* est représenté au niveau syntaxique par une trace, le nœud *trace*, qui réfère à la même entité sémantique *ent* que le syntagme nominal modifié par *facile* et représenté par le nœud *np-in*.

Si nous nous en tenons au niveau syntaxique, nous pouvons remarquer que notre proposition peut se combiner avec celle que nous avons faite pour les auxiliaires de temps et les verbes causatifs à la section 2.3 et permet de rendre compte des constructions suivantes modélisées dans HPSG par A. Abeillé, D. Godard, P. Miller et I. Sag[AGMS98] :

Cette chanson est facile à faire apprendre.
C'est le genre de gens utiles à avoir fréquenté pendant sa jeunesse.

4.5 Comparaison avec d'autres formalisations de la sémantique

Si l'on ne considère que le niveau sémantique dans les grammaires d'interaction synchrones, la modélisation que nous proposons s'inscrit dans une tendance qui est de plus en plus répandue actuellement et qui est de mettre en avant des représentations sémantiques sous-spécifiées [vDP96]. La motivation principale en est que les ambiguïtés de portée de quantificateurs et de connecteurs peuvent entraîner une explosion des lectures possibles d'une phrase et qu'il est utile de pouvoir toutes les représenter à la fois sous une forme compacte. Une autre motivation est qu'en traduction il n'est souvent pas nécessaire

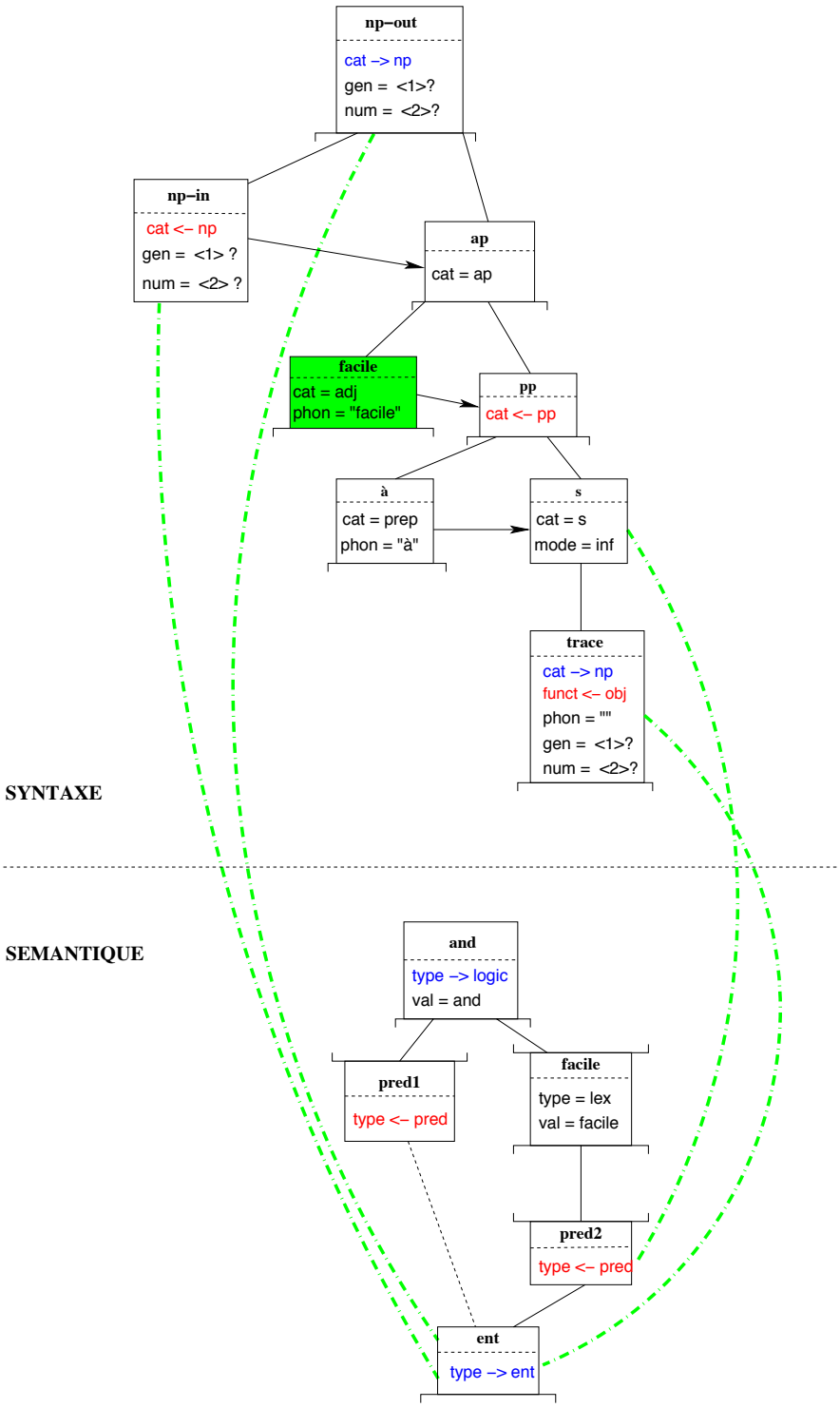


FIG. 4.19 – représentation syntaxico-sémantique de *facile (à)*

de résoudre toutes les ambiguïtés d'un énoncé pour le traduire dans une autre langue.

Ce n'est pas un hasard si le projet Verbmobil a donné lieu à tout un travail de recherche autour de la notion de représentation sémantique sous-spécifiée. Ce projet [Wah93] visait à faire dialoguer des hommes d'affaires allemands et japonais à travers l'anglais comme langue utilisée en réception. Le système devait donc être capable de traduire des dialogues d'allemand et de japonais en anglais. Autour de ce projet, un certain nombre de formalismes de représentation sémantique ont été développés qui sont très voisins et qui tous mettent en avant la notion de sous-spécification : l'*Underspecified Discourse Representation Theory (UDRT)* [Rey93], la *Hole Semantics* [Bos95] qui est le point de départ du langage LUD [BBL$^+$96] et la *Minimal Recursion Semantics (MRS)* [CFS99].

Certains de formalismes apparaissent comme une version sous-spécifiée d'une représentation sémantique particulière : l'UDRT est liée à la DRT et la MRS à la logique des prédicats. D'autres, comme la *Hole Semantics* apparaissent plutôt comme un méta-formalisme pouvant s'appliquer à différents formalismes sémantiques objets comme la DRT, la logique des prédicats ou la logique dynamique.

L'idée commune de départ est de transformer les structures sémantiques arborescentes en ensembles plats de prédicats étiquetés. C'est pourquoi on qualifie ce type de représentation de sémantique plate. Des relations prédicat-arguments entre étiquettes permettent de reconstituer les structures sémantiques complexes correspondantes.

En présentant les choses ainsi, on se donne la possibilité de sous-spécifier les représentations à l'aide de relations de domination entre étiquettes : en indiquant qu'une étiquette l_1 domine une étiquette l_2, on va pouvoir exprimer que le ou les prédicats attachés à l_2 sont dans le champ d'un des arguments d'un prédicat attaché à l_1. Ces relations de domination sont utilisées pour exprimer les ambiguïtés de portée des quantificateurs, des adverbes, des conjonctions de coordination et autres.

On retrouve sous une autre forme les descriptions de DAG sémantiques des grammaires d'interaction. Néanmoins, les représentations s'en éloignent parfois sur certains points : unicité de la racine, attachement des étiquettes uniquement à des prédicats et pas à des individus, attachement d'une étiquette à une conjonction de prédicats, celle-ci étant implicite. En plus, au delà d'un noyau commun, chaque formalisme a sa spécificité.

Ainsi dans la *Hole Semantics*, les étiquettes sont soient des constantes, soit des trous et c'est le mécanisme d'identification des trous par des constantes qui va fournir le moyen de réduire la sous-spécification et de produire les modèles d'une description donnée. Une constante ne pouvant être utilisée qu'une seule fois pour remplir un trou, nous ne sommes pas loin du mécanisme de neutralisation des polarités des grammaires d'interaction : il suffit de remplacer un trou par une polarité négative et une constante par une polarité positive.

Dans la MRS, il n'y a aucun mécanisme permettant d'obtenir les modèles d'une description. Par contre, elle peut être utilisée dans une grammaire pour permettre la construction de la représentation sémantique d'une phrase en parallèle avec sa composition syntaxique. Deux règles de composition sémantique

sont utilisées pour cela : la combinaison intersective et la combinaison de portée qui permettent d'obtenir la représentation sémantique d'un syntagme à partir de celles de ses constituants immédiats. Dans ces règles, une étiquette représentant la racine locale de la représentation sémantique d'un syntagme joue un rôle particulier. La combinaison intersective permet de réaliser la conjonction de représentations sémantiques alors que la combinaison de portée permet d'inscrire une représentation sémantique dans la portée d'un prédicat présent dans une seconde représentation.

Comme langage de représentation sémantique sous-spécifiée, nous pouvons encore mentionner *Constraint Language for Lambda-Structures (CLLS)* [ENRX98] qu'on peut résumer comme une version sous-spécifiée de la sémantique de Montague car il est fondé sur la description de λ-termes sous-spécifiés.

4.6 Comparaison avec d'autres approches de l'interface syntaxe-sémantique

Si on examine maintenant la façon dont les formalismes linguistiques combinent syntaxe et sémantique, le modèle des grammaires d'interaction synchrones évoquent immédiatement celui des TAG synchrones [SS90]. Historiquement, ce n'est pas la première tentative d'associer une représentation sémantique aux TAG et ce n'est pas non plus la façon la plus répandue.

Dans la vision la plus courante, on part des arbres de dérivation qui représentent la dépendance entre les opérations d'adjonction et de substitution utilisées pour réaliser une analyse et on considère que ces arbres reflètent les dépendances sémantiques entre les différents lexèmes [VS87]. Ce postulat fonctionne dans de nombreux cas mais un certain nombre de contre-exemples viennent l'ébranler d'où des travaux pour faire évoluer les TAG vers un formalisme où les opérations de composition syntaxique soient plus en harmonie avec la sémantique [RVSW95].

Les TAG synchrones utilisent quant à eux deux grammaires TAG, une grammaire syntaxique et une grammaire sémantique, reliées par un mécanisme de synchronisation. Dans la présentation initiale [SS90], la synchronisation s'effectue à travers les arbres dérivés, ce qui augmente le pouvoir d'expression du formalisme tout en le rendant plus difficile à traiter automatiquement. S. Shieber y remédie en proposant d'assurer la synchronisation à travers un isomorphisme entre arbres de dérivation [Shi94]. Le mécanisme de synchronisation est néanmoins trop rigide pour exprimer des phénomènes tels que la portée des quantificateurs, d'où des propositions pour relâcher le mécanisme [RS96].

Par rapport aux grammaires d'interaction, les propositions d'intégration de la sémantique aux TAG présentent le défaut d'un lien trop étroit et trop rigide entre syntaxe et sémantique. Toute opération de composition sémantique doit s'accompagner d'une opération équivalente au niveau syntaxique, alors que dans les grammaires d'interaction, les différentes interprétations liées à différentes priorités quant à la portée des quantificateurs s'obtiennent par des neutralisations de traits au niveau sémantique qui ne touchent pas du tout le niveau syntaxique. En plus, pour ce qui est des TAG synchrones, le formalisme relativement compliqué ne facilite pas l'écriture des grammaires et

l'implémentation.

C. Gardent et L. Kallmeyer proposent une nouvelle vision de l'interface syntaxe sémantique pour les TAG [GK03] qui utilise la *Hole Semantics* de J. Bos [Bos95] et qui s'appuie sur les arbres dérivés pour assurer la synchronisation avec la syntaxe. Des traits particuliers dans ces arbres représentent des étiquettes prédicatives et des entités sémantiques. Dans le processus de composition syntaxique, l'unification de ces traits va contribuer à la construction en parallèle de la représentation sémantique. En fin d'analyse, la représentation sémantique peut rester sous-spécifiée et c'est le remplissage des trous qui va permettre d'obtenir les différentes interprétations sémantiques correspondantes. Cette approche beaucoup plus souple que celles précédemment exposées permet d'en dépasser certaines limites et elle se rapproche beaucoup de notre formalisation. La principale différence se situe au niveau du formalisme syntaxique utilisé mais il y a d'autres différences moins importantes au niveau purement sémantique :

- le mécanisme de réduction de la sous-spécification utilise le remplissage de trous au lieu de la neutralisation de traits mais, comme nous l'avons déjà dit, les deux opérations sont équivalentes ;
- les individus sémantiques ne sont pas intégrés comme les prédicats en tant que nœuds de la structure sémantique ;
- les conjonctions de prédicats sont implicitement représentées en associant la même étiquette à plusieurs prédicats alors dans nos DAG sémantiques un nœud ne représente qu'un seul prédicat.

De façon indépendante à nos travaux et en partant des TAG, R. Muskens a imaginé un formalisme syntaxique fondé sur les descriptions d'arbres polarisés [Mus01] qui est très proche des grammaires d'interaction primitives, telles qu'elles ont été présentées à la section 1.5. La sémantique y est intégrée par un étiquetage des nœuds syntaxiques à l'aide de λ-termes à la Montague. Malheureusement, cette intégration présente le même défaut que celui déjà mentionné pour les grammaires catégorielles qui est de devoir alourdir artificiellement la syntaxe pour représenter les complications de la sémantique, notamment les ambiguïtés de portée des quantificateurs.

Dans un autre ordre d'idées, on trouve des similitudes entre notre approche et celle de S. Kahane [Kah02]. Suivant la théorie Sens-Texte [Mel88], il propose le formalisme de la Grammaire d'Unification Sens-Texte qui comporte 3 niveaux : sémantique, syntaxique et phonologique. Les structures représentées au niveau sémantique et syntaxique sont respectivement des DAG et des arbres et la correspondance entre les deux se fait par ensemble de nœuds car les nœuds représentent des mots et non des syntagmes. Le fait que les nœuds représentent des mots rend moins nécessaire la représentation de la sous-spécification que dans une approche syntagmatique car les dépendances non bornées sont représentées par des relations de parenté dans l'arbre de dépendance. Par contre, cela pose des problèmes pour la représentation de dépendances comme celle que l'on trouve avec les propositions relatives, d'où la nécessité d'ajouter des mécanismes spécifiques [Kah97].

Chapitre 5

L'organisation modulaire et hiérarchisée des grammaires d'interaction

Un formalisme linguistique, même s'il ne se confond pas avec une théorie linguistique, n'est pas non plus neutre par rapport à toute théorie linguistique. Il fait, même si ce n'est pas toujours explicite, des hypothèses sur la langue. L'évaluation de la pertinence du formalisme procède de la même démarche expérimentale qui permet au physicien de vérifier ses hypothèses. La matière expérimentale est ici constituée des corpus réels de la langue. Cela nécessite de construire des grammaires électroniques à large couverture qui puissent être ensuite utilisables par des analyseurs. Or, qui dit grammaire à large couverture, dit grammaire de taille énorme. Toutes les approches linguistiques formelles sont confrontées à ce problème. La factorisation de cette information y est cruciale pour différentes raisons :

- Ces grammaires doivent rester lisibles et pour cela elles doivent intégrer au maximum la généralisation que permet la connaissance linguistique. Idéalement, un linguiste doit pouvoir via des interfaces appropriées interagir avec de telles grammaires. Il doit pouvoir les construire, les mettre à jour et les modifier en fonction de ses choix linguistiques.
- A tout moment, ces grammaires doivent rester cohérentes. Toute modification ponctuelle ne doit pas remettre en cause l'ensemble de l'échafaudage. Il faut pouvoir mettre à jour, faire évoluer ces grammaires sans devoir à chaque fois repartir de zéro.
- Enfin cette factorisation doit rendre plus efficace l'extraction d'informations de ces grammaires pour l'analyse syntaxique et sémantique.

Les formalismes grammaticaux basés sur l'unification utilisent de façon naturelle la relation de subsumption entre structures de traits comme mécanisme d'héritage pour structurer les grammaires en hiérarchies de modules. Ces hiérarchies peuvent en plus être contrôlées par un typage des structures de traits [Car92]. Ce mécanisme d'héritage est souvent combiné avec des règles lexicales qui expriment des transformations de configurations syntaxiques canoniques. La meilleure illustration de cette organisation est donnée par HPSG [PS94].

Alors que HPSG fournit un cadre abstrait de représentation des lexiques, R. Evans et G. Gazdar se préoccupent de l'implémentation de ces lexiques en proposant un langage de représentation lexicales qu'ils ont baptisé DATR [EG96]. Ce langage est applicable à tout système hiérarchique fondé sur l'héritage non monotone et est particulièrement adapté aux grammaires d'unification. Cependant, il permet aussi de coder des lexiques fondés sur les grammaires d'arbres. En particulier, R. Evans, G. Gazdar and D. Weir ont proposé d'utiliser le langage DATR pour organiser les TAG [EGW95]. Toutes les informations syntaxiques, même la topologie des arbres et les règles lexicales, sont codées sous forme d'une hiérarchie de structures de traits. Ce système est aussi utilisé dans le projet LexSys [CNS+98] pour représenter de façon compacte les LDTG (Lexicalized Description Tree Grammars) [RVSW95].

La notion de description d'arbre de par sa souplesse ouvre d'autres perspectives pour la représentation modulaire des grammaires. Elle a notamment été utilisée pour la représentation des grammaires anglaise et chinoise du système XTAG[XTA01]. M.-H. Candito a exploité cette notion pour pousser plus à fond la factorisation dans l'organisation de grammaires TAG du français et de l'italien [Can99]. Elle définit ce qu'elle appelle une méta-grammaire, c'est-à-dire une structure qui préexiste à une grammaire et qui va pouvoir l'engendrer. Une méta-grammaire est organisée suivant plusieurs dimensions en autant de hiérarchies de classes. Chaque classe représente une construction grammaticale particulière sous forme d'une description d'arbres. Les différentes dimensions représentent une façon particulière de découper une grammaire en niveaux relativement indépendants. Par exemple, pour le verbe, M.-H. Candito distingue 3 dimensions : celle de la sous-catégorisation initiale qui donne les arguments du verbe dans sa forme canonique, celle de la redistribution des fonctions syntaxiques et enfin celle de la réalisation des fonctions syntaxiques finales. Les arbres TAG élémentaires qui vont constituer la grammaire proprement dite sont engendrés automatiquement par croisement des classes terminales des différentes dimensions. Ce croisement n'est pas complètement libre mais contrôlé par des contraintes positives ou négatives. C'est cette notion de croisement liée à celle de dimension qui constitue l'originalité du travail de M.-H. Candito.

Une difficulté dans le système présenté par M.-H. Candito est que le mécanisme qui assure la composition tant verticale (par l'héritage) qu'horizontal (par le croisement) des classes de la méta-grammaire manque d'homogénéité et fait appel à des procédés ad hoc qui se prêtent mal à la formalisation et à la généralisation :

- La présentation de la hiérarchie de classes n'est pas purement déclarative. A certaines classes qui décrivent les redistributions de fonctions syntaxiques, sont associées des actions qui révisent les valeurs des traits représentant ces fonctions. En conséquence, lorsqu'une classe hérite de plusieurs de celles-ci, l'ordre dans lequel se réalise l'héritage n'est plus indifférent. C'est la même chose pour l'ordre dans lequel s'effectue le croisement de classes terminales.
- Lorsque deux classes sont composées par héritage ou par croisement, il y a 3 façons de réaliser l'identification de nœuds entre les deux classes : l'utilisation de noms globaux à la hiérarchie, l'utilisation de valeurs communes du trait *fonction* et enfin le recours à la notion de tête.

– Les contraintes qui imposent des restrictions au croisement des classes terminales sont exprimées de 4 façons possibles : principes de bonne formation, classes disjonctives, relations binaires de co-occurrence positive et négatives. Et, malheureusement, cet arsenal n'est pas suffisant pour exprimer toutes les contraintes de croisement, celles par exemple liées au croisement de plusieurs classes réalisant différentes fonctions syntaxiques.

B. Gaiffe, B. Crabbé et A. Roussanaly [GCR02] ont développé un compilateur de méta-grammaires qui corrigent certains de ces défauts, notamment l'absence de monotonie et les limitations liées aux dimensions.

Les idées de M.-H. Candito, si elles ont été appliquées aux TAG, n'en sont pas pour autant liées à ce formalisme et peuvent s'appliquer à tout autre formalisme linguistique, pour peu qu'il puisse être exprimé à l'aide de descriptions d'arbres. C'est le cas des grammaires d'interaction et nous proposons un essai pour une organisation modulaire et hiérarchisée des grammaires d'interaction qui reprend l'essentiel de la proposition de M.-H. Candito : les constructions syntaxiques caractéristiques d'une grammaire sont représentées sous forme de descriptions d'arbres et organisées en une hiérarchie de classes par une relation d'héritage ; les classes terminales de la hiérarchie doivent ensuite être croisées pour produire toutes les constructions syntaxiques figurant dans le lexique associé. Plutôt que de parler de méta-grammaire, nous utiliserons le terme "grammaire hiérarchisée" pour désigner cette structure.

La proposition qui est décrite dans ce chapitre n'est qu'une première proposition qui demande à être enrichie. Sur certaines questions, nous faisons, en première approximation, le choix de la simplicité et nous en laissons d'autres ouvertes :

– nous proposons une représentation des classes purement déclarative et une stricte monotonie de la relation d'héritage[18] ;
– lorsque l'on compose deux classes, l'identification entre certains de leurs nœuds est réalisée par le biais de noms globaux à la hiérarchie ;
– nous ne conservons pas de distinctions explicites entre plusieurs dimensions et nous proposons un mécanisme uniforme de croisement entre classes terminales qui soit contrôlé par deux types de relations : pour une classe dite *disjonctive*, les sous-classes immédiates ne pourront pas être croisées et pour une classe *conjonctive*, toutes ses sous-classes immédiates devront être croisées.

Enfin, ce qui est tout à fait nouveau par rapport à la proposition de M.-H. Candito, nous nous attachons à définir un mécanisme relativement souple d'ancrage lexical des classes résultant du croisement de classes terminales. Par ancrage lexical, nous entendons l'opération qui consiste à lier les classes aux mots de la langue. Chez M.-H. Candito, cet ancrage se fait de façon relativement rigide. Les arbres TAG engendrés par une méta-grammaire sont regroupés en familles suivant des sous-catégorisations initiales types. Un lexique associe ensuite les mots de la langue avec des familles d'arbres. Nous proposons d'associer une structure de traits à chaque classe de la grammaire, structure de traits que nous appellerons le profil morpho-syntaxique de la classe et qui décrira les pro-

[18]Nous proposerons une légère entorse à la monotonie avec le renommage possible des nœuds.

priétés morpho-syntaxiques des mots pouvant ancrer celle-ci. Les mots ayant aussi un profil se présentant comme une structure de traits, le lien entre une classe et un mot va s'effectuer par unification des profils correspondants. Indépendamment de nous, B. Gaiffe, B. Crabbé et A. Roussanaly [GCR02] ont introduit la même notion de profil dans leur compilateur de méta-grammaires.

Nous nous cantonnerons au niveau syntaxique des grammaires d'interaction mais nos propositions s'étendent à l'ensemble des grammaires d'interaction synchrones combinant syntaxe et sémantique. Par ailleurs, ces propositions ne sont en rien dépendantes du formalisme particulier des grammaires d'interaction et peuvent être reprises pour tout formalisme linguistique basé sur les descriptions d'arbres ou de DAG.

5.1 Le mécanisme d'héritage et de croisement de classes grammaticales

5.1.1 Définition d'une grammaire d'interaction hiérarchisée

Définition 5.1.1. *Une* grammaire d'interaction hiérarchisée *est un ensemble fini de descriptions syntaxiques sur des environnements disjoints appelées classes. Cet ensemble de classes est structuré de la façon suivante :*

- *une relation binaire entre classes qui a la signification* « hérite directement de » *définit un DAG ;*
- *chaque classe est soit ordinaire, soit conjonctive ou disjonctive.*

Chaque classe représente une construction grammaticale particulière de la langue et est désignée par un nom.

La figure 5.1 présente une grammaire hiérarchisée qui couvre presque complètement les mots interrogatifs du français (adverbes, pronoms et déterminants) à la forme interrogative directe[19]. Nous la baptiserons G_{wh}.

L'intérêt d'un tel exemple est que la grammaire est suffisamment restreinte pour qu'on puisse la présenter de façon complète et en même temps suffisamment complexe pour que l'on puisse saisir l'intérêt de l'organisation proposée. Les phrases suivantes donnent une idée de la complexité de la grammaire des interrogatives directes.

(a) *Qui* pense que Jean voit Marie ?
(b) *Qui* Jean pense-t-il que Marie voit ?
(c) *Qu'*est-ce que Jean pense que Marie aime ?
(d) *Comment* Jean pense-t-il que Marie dort ?
(e) A la fille de *quel* ami est-ce que Jean pense ?
(f) Dans la maison de *laquelle* des sœurs de Jean Marie dort-elle ?

La grammaire G_{wh} rend compte des phénomènes illustrés par ces phrases et d'autres encore. Elle couvre notamment tous les phénomènes illustrés par la suite de phrases interrogatives partielles tests du TSNLP [LORP+96] (les *S-Question-wh*). Seules les expressions figées sans verbe, telles que *« quoi de*

[19] Pour l'instant, nous ne parlerons pas de la structure de traits représentant le profil associé à chaque classe. Elle sera expliquée par la suite.

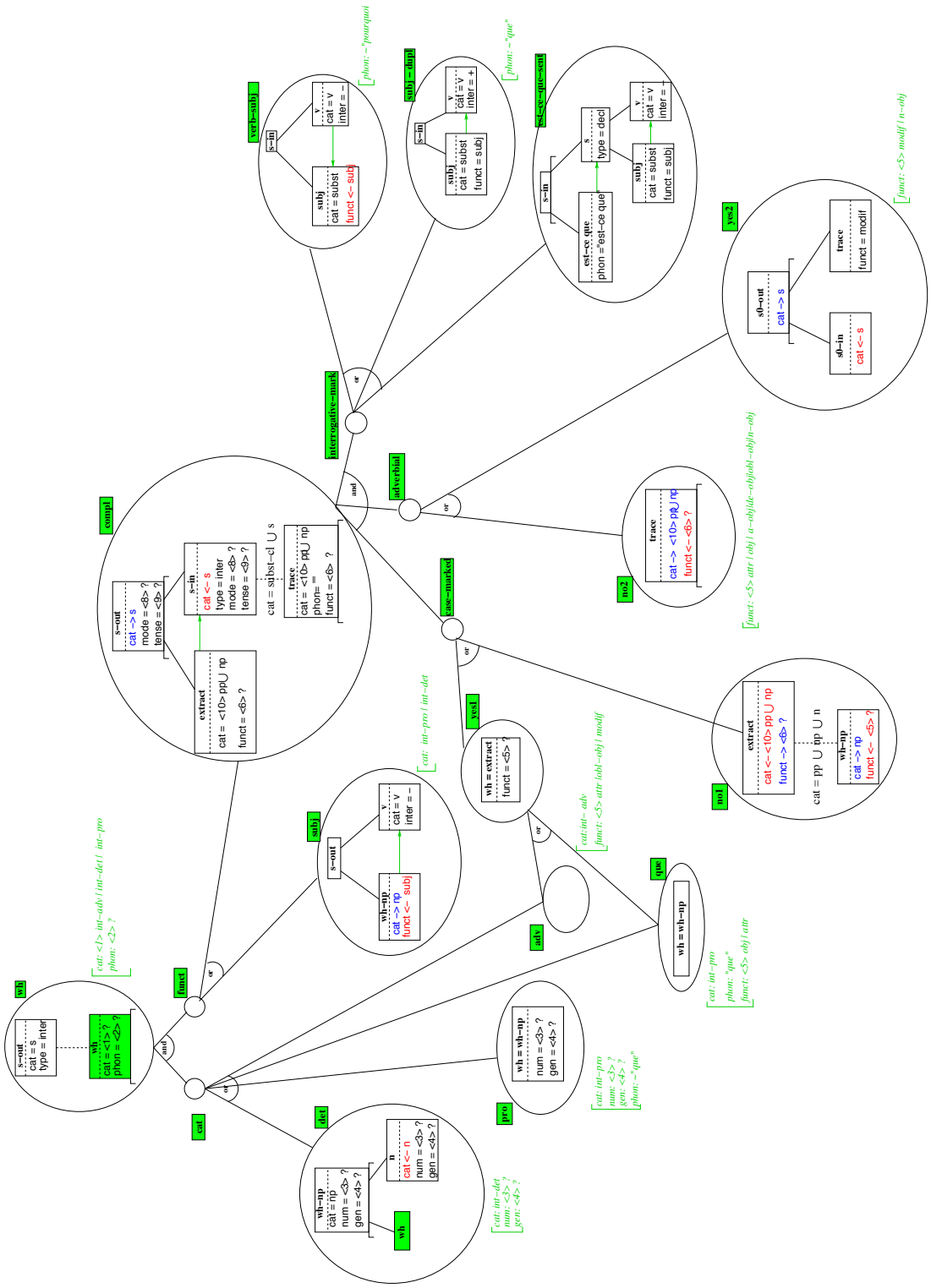


FIG. 5.1 – Grammaire d'interaction hiérarchisée G_{wh} des mots interrogatifs du français à la forme interrogative directe

neuf ? », ou avec un verbe à l'infinitif, telles que « *quoi faire ?* », ou relativement figées comme « *quel est cet homme ?* » ne sont pas prises en compte.

Expliquons l'architecture de G_{wh}. La racine *wh* se présente comme une classe conjonctive[20] pour exprimer deux dimensions d'un mot interrogatif : sa catégorie grammaticale représentée par la classe *cat* et sa fonction syntaxique représentée par la classe *funct*.

La classe *cat* est disjonctive[21], exprimant par là les différentes alternatives possibles pour la catégorie grammaticale : déterminant, pronom *que* marqué casuellement, pronom autre que *que* non marqué casuellement ou adverbe.

La classe *funct* est elle-même disjonctive, exprimant qu'elle peut se réaliser soit comme sujet (classe *subj*), soit comme complément. La classe *compl* exprime cette deuxième alternative et elle conjonctive du fait qu'elle offre trois dimensions : la première, représentée par la classe *case-marked*, précise si le mot interrogatif est marqué casuellement, la seconde, représentée par la classe *adverbial*, a trait au caractère circonstanciel ou non du complément et la dernière, représentée par la classe *interrogative-mark*, indique comment le type interrogatif est marqué (inversion verbe-sujet, redoublement du sujet, présence de *est-ce que*).

Donnons quelques précisions qui permettent de comprendre le contenu des classes. Les noms de nœuds syntaxiques ont une portée qui est celle de toute la hiérarchie alors que les variables d'environnement sont locales aux classes. Le nœud *wh* représente le mot interrogatif et, dans le cas où celui-ci est un déterminant ou un pronom, le nœud *wh-np* représente le plus petit syntagme nominal le contenant. Ainsi dans la phrase (e) ci-dessus, le nœud *wh* correspond à *quel* tandis que *wh-np* correspond à « *quel ami* ». Le nœud *s-out* représente la phrase interrogative complète. Dans le cas où *wh-np* n'est pas le sujet de la phrase interrogative, on considère que celle -ci est formée d'un syntagme extrait, représenté par le nœud *extract*, qui contient *wh-np*, suivi de la phrase sans ce syntagme représentée par le nœud *s-in*. Lorsque les dépendances non bornées en cascade sont possibles, la profondeur à laquelle *wh-np* se situe dans *extract* peut être plus ou moins grande. Nous avons deux cas avec les phrases (b) et (f) par exemple. Dans la phrase (b), l'objet extrait *extract* est *qui* tandis que *s-in* correspond à « *Jean pense-t-il que Marie voit* ». Ici, *extract* se confond avec *wh-np*. Dans (f), le syntagme *extract* est « *Dans la maison de laquelle des sœurs de Jean* » et *s-in* se réduit à « *Marie dort-elle* ».Le syntagme *extract* contient *wh-np* qui correspond à *laquelle*.

5.1.2 Calcul de l'héritage et du croisement de classes

L'héritage et le croisement de classes font tous les deux appel à la même opération de *composition de classes* que nous allons définir maintenant. Les classes étant des descriptions syntaxiques, nous allons définir la composée de deux descriptions syntaxiques.

[20] Cette propriété est marquée par un arc de cercle étiqueté *and* reliant tous les liens entre la classe en question et ses fils.

[21] Cette propriété est marquée par un arc de cercle étiqueté *or* reliant tous les liens entre la classe en question et ses fils.

Définition 5.1.2. *Soit deux descriptions syntaxiques D_1 et D_2 définies sur deux environnements disjoints Γ_1 et Γ_2*[22]*. La composée de D_1 et D_2 est une description D qui est définie ainsi :*

- $|D| = |D_1| \cup |D_2|$;
- *son environnement associé est* $\Gamma = \Gamma_1 \uplus \Gamma_2$;
- *pour tout nœud N de $|D|$, $Feat_D(N)$ est la somme des structures de traits $Feat_{D_i}(N)$ pour $i \in \{1,2\}$ où ces structures sont définies*[23] ;
- *les relations sur $|D|$ sont obtenues en faisant l'union des relations correspondante sur $|D_1|$ et $|D_2|$.*

A l'aide de cette opération de composition et de la relation d'héritage, nous pouvons définir le *contenu complet* d'une classe.

Définition 5.1.3. *Soit une grammaire d'interaction hiérarchisée. Soit dans cette grammaire une classe D_0 qui hérite directement des classes $D_1, \ldots, D_n$. Le* contenu complet *de cette classe est la description notée $Compl(D_0)$ qui est définie récursivement comme la composée de $Compl(D_1), \ldots, Compl(D_n)$*

La figure 5.2 nous montre le contenu complet de la classe *wh.funct.compl.case-marked.no1* qui résulte de la composition verticale des contenus spécifiques des classes *wh*, *funct*, *compl*, *case-marked* et *no1*.

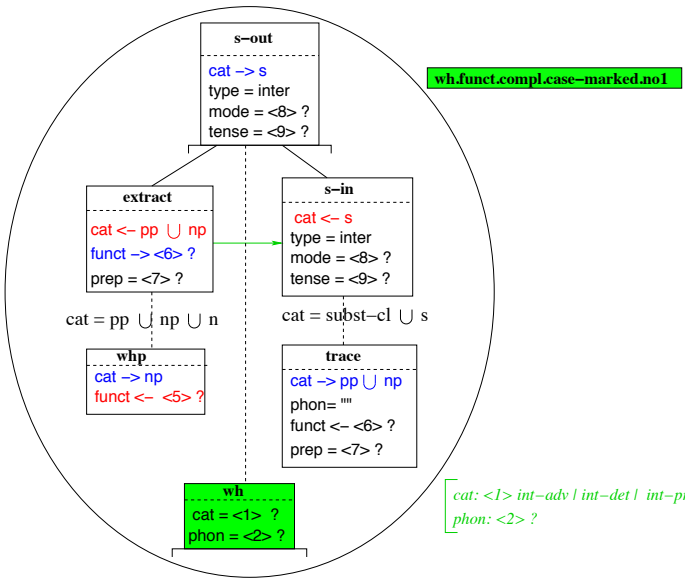


FIG. 5.2 – Contenu complet de la classe terminale *wh.funct.compl.case-marked.no1* de la grammaire G_{wh}

Le contenu complet des 11 classes terminales de la hiérarchie de notre grammaire ne nous fournit pas encore les descriptions qui vont être associés aux

[22] Si les environnements ne sont pas disjoints, on renomme certaines de leurs variables pour qu'ils le soient.

[23] La somme de deux structures de traits ayant été définie lorsque celles-ci portent sur le même environnement, nous considérerons que les structures de traits $Feat_{D_i}(N)$ portent toutes sur Γ.

mots interrogatifs dans un lexique. Ces classes doivent encore être composées entre elles mutuellement de la même façon que l'on compose les classes le long d'une branche d'héritage, la seule différence est que la composition est maintenant horizontale et non plus verticale. Pour différencier la composition horizontale de la composition verticale, nous parlerons de *croisement* des classes terminales à propos de la première. La composition verticale est contrôlée par la relation d'héritage, alors que la composition horizontale l'est par les caractères disjonctif ou conjonctif des classes : une classe disjonctive va empêcher les croisements de ses sous-classes directes et de celles qui en héritent tandis qu'une classe conjonctive va forcer ce croisement. On pourrait théoriquement se passer des croisements en les déclarant explicitement dans la relation d'héritage mais ce serait au prix d'un alourdissement de la hiérarchie car la combinatoire du nombre de croisements peut être élevée.

Il reste à définir précisément l'opération de croisement à travers son résultat qu'on appellera une *classe lexicale*, c'est-à-dire une classe qui va nous fournir une description qui va pouvoir être associé à un mot dans un lexique.

Définition 5.1.4. *Une classe lexicale est obtenu par croisement d'un ensemble E de classes terminales complètes tel que toute classe C de la grammaire dont hérite un élément de l'ensemble E a la propriété suivante :*

- *si C est une classe disjonctive, une seule des sous-classes immédiates de C est un ancêtre d'un élément de E ;*
- *si C est une classe conjonctive, toutes les sous-classes immédiates de C sont des ancêtres d'éléments de E.*

De cette façon, nous obtenons 22 classes lexicales énumérées dans le tableau 5.1[24] illustrées chacune par un exemple.

Détaillons une de ces classes lexicales, la classe *det.no1.no2.verb-subj* qui résulte du croisement de *det*, *no1*, *no2* et *verb-subj*. La figure 5.3 nous en donne le contenu.

5.1.3 Problème de la portée des noms de nœuds syntaxiques

Nous avons indiqué que les noms de nœuds syntaxiques présents dans les descriptions formant le contenu des classes ont une portée qui est celle de l'ensemble de la hiérarchie. Or, cela peut être un inconvénient. Considérons par exemple une grammaire hiérarchisée des verbes à deux compléments indirects introduits respectivement par les prépositions *à* et *de*. Simplifions cette grammaire en la réduisant à un classe *verb* qui contient les propriétés syntaxiques générales du verbe et deux classes *a-compl* et *de-compl* qui introduisent les deux compléments. La classe terminale représentant les verbes à deux compléments, baptisée *double-compl*, héritera de ces 3 classes. On peut remarquer que les deux classes *a-compl* et *de-compl* diffèrent uniquement par leurs prépositions donc il serait souhaitable de factoriser leur partie commune en une classe unique *ind-compl* dont elles hériteraient toutes les deux. Malheureusement, du fait de la portée globale des noms de nœuds, les deux compléments vont se trouver confondus. Une solution consiste à autoriser le renommage des nœuds

[24]Une classe lexicale est désignée par suite des classes terminales qui ont été croisées pour la construire séparées par un point.

classe	exemple illustrant le classe
det.subj	**combien de** gars arrivent ?
det.no1.no2.verb-subj	**quelle** pomme mange Pierre ?
det.no1.no2.subj-dupl	**quelle** pomme Pierre mange-t-il ?
det.no1.no2.est-ce-que-sent	**quelle** pomme est-ce que Pierre mange ?
det.no1yes2.verb-subj	dans **quel** bureau travaille Pierre ?
det.no1yes2.subj-dupl	dans **quel** bureau Pierre travaille-t-il ?
det.no1yes2.est-ce-que-sent	dans **quel** bureau est-ce que Pierre travaille ?
pro.subj	**lequel** travaille avec Pierre ?
pro.no1.no2.verb-subj	**qui** est cet homme ?
pro.no1.no2.subj-dupl	**qui** Pierre aime-t-il ?
pro.no1.no2.est-ce-que-sent	**qui** est-ce que Pierre aime ?
pro.no1.yes2.verb-subj	dans **lequel** des bureaux travaille Pierre ?
pro.no1.yes2.subj-dupl	dans **lequel** des bureaux Pierre travaille-t-il ?
pro.no1.yes2.est-ce-que-sent	dans **lequel** des bureaux est-ce que Pierre travaille ?
que.no2.verb-subj	**que** fait Pierre ?
que.no2.est-ce-que-sent	**qu'**est-ce que Pierre fait ?
adv.no2.verb-subj	**comment** est Pierre ?
adv.no2.subj-dupl	**comment** Pierre est-il ?
adv.no2.est-ce-que-sent	**comment** est-ce que Pierre est ?
adv.yes2.verb-subj	**où** travaille Pierre ?
adv.yes2.subj-dupl	**où** Pierre travaille-t-il ?
adv.yes2.est-ce-que-sent	**où** est-ce que Pierre travaille ?

TAB. 5.1 – Liste des classes lexicales de la grammaire G_{wh}

dans l'héritage. C'est ce que présente la figure 5.4. La classe *a-compl* hérite de la classe *ind-compl* en renommant le nœud *ind-obj* en *a-obj*. De même, la classe *de-compl* hérite de la classe *ind-compl* en renommant le nœud *ind-obj* en *de-obj*. Ensuite, dans la classe *double-compl*, les deux nœuds représentant les deux compléments portant des noms différents peuvent être distingués sans problèmes.

5.2 Ancrage lexical des classes à l'aide de structures de traits

Une grammaire d'interaction hiérarchisée n'est pas lexicalisée et ne peut pas être utilisée directement pour l'analyse syntaxique. Les classes lexicales qui sont engendrées automatiquement par croisement des classes terminales de la grammaire ne sont pas rattachées à des mots de la langue. Le mécanisme qui va permettre de lier les classes lexicales aux mots dans un lexique utilise la notion de profil morpho-syntaxique associé à une classe.

5.2.1 Profil morpho-syntaxique d'une classe, profil morpho-syntaxique d'un mot

A chaque classe d'une grammaire hiérarchisée, nous associons son *profil morpho-syntaxique* dont le but est de décrire les propriétés morpho-syntaxiques des mots susceptibles d'ancrer cette classe. C'est une structure de traits de la même nature que celles qui sont attachées aux nœuds des descriptions syntaxiques, avec la restriction que les traits sont tous neutres.

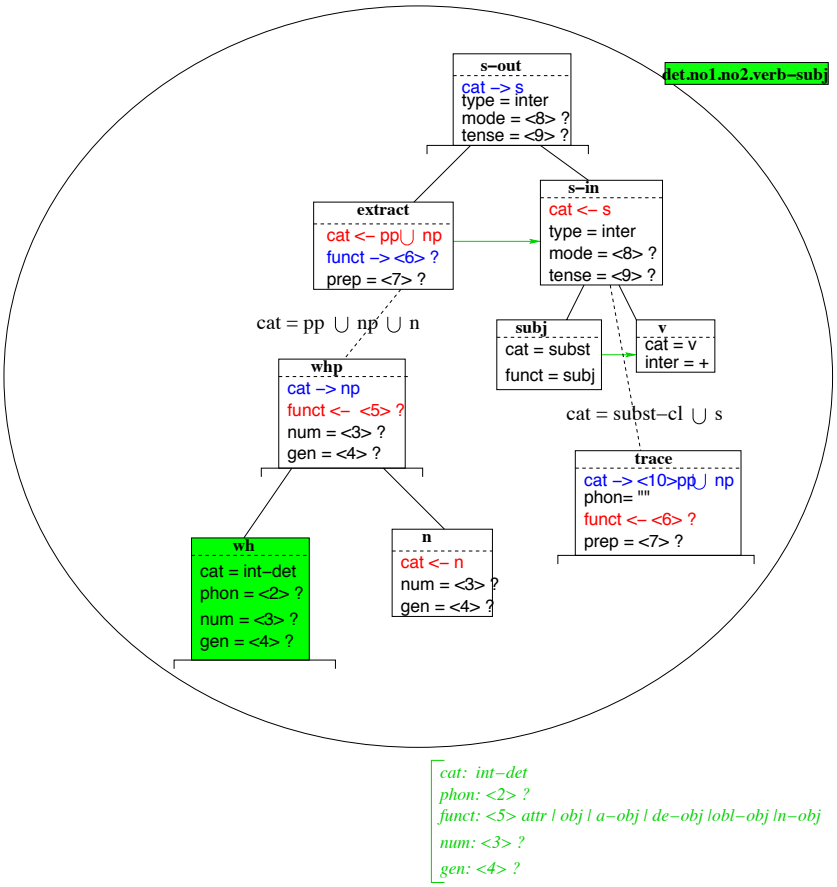


FIG. 5.3 – Contenu complet de la classe lexicale *det.no1.no2.verb-subj* de la grammaire G_{wh}

De la même façon que la relation d'héritage permet de calculer le contenu complet d'une classe par composition, elle permet de calculer le profil morpho-syntaxique complet d'une classe par unification des profils spécifiques des classes dont elle hérite avec le sien. Prenons l'exemple de la classe *det* de la grammaire G_{wh}. Sur la figure 5.1, le profil spécifique de chaque classe est affiché à côté de celle-ci ; lorsque rien n'est affiché, cela signifie que le profil associé est la structure de traits indéterminée. Pour *det*, le profil complet résulte de l'unification de deux profils spécifiques, le sien et celui de *wh* et il se présente donc ainsi :

$$\begin{bmatrix} cat : int - det \\ phon :< 2 >? \\ num :< 3 >? \\ gen :< 4 >? \end{bmatrix}$$

Le processus de croisement ne diffère pas dans son contenu de celui de l'héritage donc au niveau des profils, il se traduit sous forme d'unification. La figure 5.3 nous donne le profil de la classe *det.no1.no2.verb-subj* qui résulte de l'unification des profils complets des 3 classes la composant.

Implicitement, nous admettons que les traits composant un profil sont com-

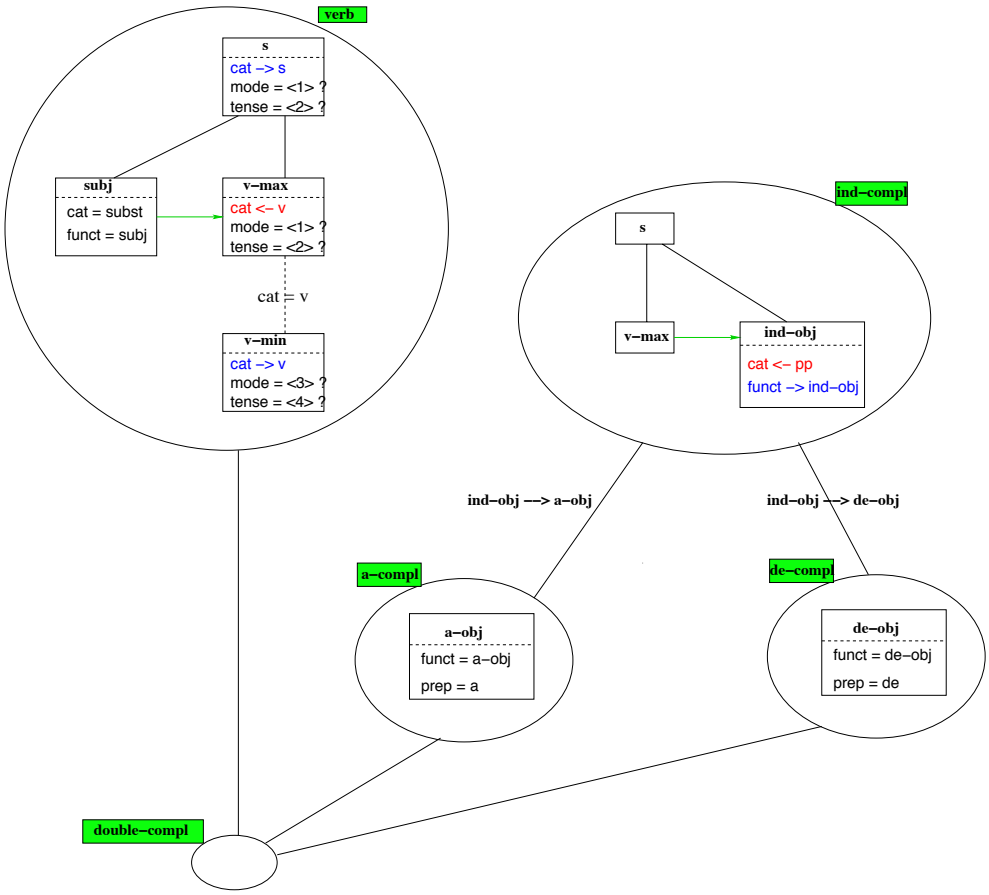


FIG. 5.4 – Grammaire simplifiée de verbes à double complémentation

plètement indépendants mais c'est loin d'être le cas dans la réalité. Supposons que le profil d'un verbe puisse comporter un trait *aux* avec comme valeurs atomiques possibles *être* et *avoir* pour indiquer l'auxiliaire avec lequel il s'emploie au passé composé et supposons qu'il ait un autre trait *trans* à valeurs booléennes qui indique si le verbe est transitif ou non. Ces deux traits sont évidemment liés par la règle qui veut que si $trans = true$ alors $aux = avoir$. Nous ne chercherons pas à modéliser cette dépendance et dans la suite nous considérerons en première approximation que les traits sont totalement indépendants même si cela entraîne une certaine redondance d'information.

De la même façon que nous avons attaché des profils morpho-syntaxiques aux classes d'une grammaire, nous allons en attacher aux mots de la langue qui auront exactement la même forme et la même signification. Ils décriront les propriétés morpho-syntaxiques des mots auxquels ils sont attachés et comme pour les classes, nous considérerons que les traits qui les composent sont totalement indépendants. Cette liaison va se faire à travers des lexiques qui se présentent comme des ensembles de couples (mot, profil morpho-syntaxique). Donnons une idée de ce que pourrait être un tel lexique en nous cantonnant à quelques mots interrogatifs. Le tableau 5.2 en donne un aperçu.

quelle	$cat = int{-}det$ $num = sg$ $gen = f$ $phon =$ "*quelle*"
qui	$cat = int{-}pro$ $num = sg$ $gen = m$ $phon =$ "*qui*"
que	$cat = int{-}pro$ $funct = attr \cup obj$ $phon =$ "*que*"
quoi	$cat = int{-}pro$ $funct = a{-}obj \cup de{-}obj \cup obl{-}obj \cup modif$ $phon =$ "*quoi*"
comment	$cat = int{-}adv$ $funct = attr \cup obl{-}obj \cup modif$ $phon =$ "*comment*"

TAB. 5.2 – Lexique partiel des mots interrogatifs du français

5.2.2 Le problème de la sélection des classes lexicales à l'aide du profil morpho-syntaxique d'un mot

Le profil de chacun des mots va servir à sélectionner de la grammaire les classes lexicales appropriées par comparaison avec les profils des diverses classes lexicales. Quelle relation va-t-on utiliser pour effectuer la comparaison entre le profil P_1 d'un mot et le profil P_2 d'une classe lexicale ? Pour prendre deux extrêmes, va-t-on utiliser la propriété pour P_1 et P_2 d'être unifiables (*unifiabilité*) ou la propriété pour P_1 d'être filtrable par P_2 (*filtrabilité*) ? Voyons quelles sont les conséquences de l'un ou l'autre de ces choix. Considérons un trait $f = v_2$ présent dans P_2 et envisageons deux cas, selon que f est présent ou non dans P_1.

– *f n'est pas présent dans P_1.*
Si nous choisissons l'unifiabilité, cela équivaut au fait qu'il l'est avec la valeur indéterminée, c'est-à-dire à la présence du trait $f =$? dans P_1. Si nous choisissons la filtrabilité, cela signifie au contraire que f est présent avec la valeur $\emptyset$, c'est-à-dire que le trait $f = \emptyset$ est présent dans P_1. Ainsi dans le profil de *quel*, il n'y a pas de trait *funct* et l'on souhaite que cela soit interprété comme équivalent à la présence d'un trait *funct* $=?$, ce qui pencherait en faveur de l'unifiabilité. Imaginons à l'inverse que l'on exprime dans le profil d'un verbe l'existence et les catégories grammaticales de ses compléments par des traits *obj, a-obj, de-obj...* Nous voudrions alors indiquer qu'un verbe intransitif est caractérisé dans son profil par l'absence d'un trait *obj*, ce qui incite à choisir plutôt la filtrabilité.

– *f est présent dans* P_1 *avec la valeur* v_1.
Si nous choisissons l'unifiabilité, la sélection réussira si v_1 et v_2 sont unifiables et si nous choisissons la filtrabilité, la sélection réussira si $v_1 \leq v_2$. Dans ce cas, l'unifiabilité semble préférable à la filtrabilité car souvent les traits morpho-syntaxiques des mots sont ambigus et c'est le contexte syntaxique de leur usage qui permet de lever l'ambiguïté. Par exemple, le trait *funct* de *quoi* a la valeur *a-obj* ∪ *de-obj* ∪ *obl-obj* ∪ *modif* mais dans le contexte *« dans quoi travaille Jean ? »* il a la valeur *modif* qui s'obtient par unification du profil de *quoi* avec celui du classe lexical *proyes2.verb-subj*.

Même si l'unifiabilité semble présenter plus d'avantages que la filtrabilité, nous n'avons pas de point de vue définitif sur la question. Dans la suite, nous choisirons l'unifiabilité. Une classe lexicale de la grammaire sera donc sélectionnée si son profil morpho-syntaxique s'unifie avec celui du mot concerné.

5.2.3 Croisement des classes terminales puis sélection ou sélection puis croisement

Telles que les choses ont été présentées jusqu'à maintenant, nous effectuons tous les croisements possibles des classes terminales d'une grammaire pour produire les classes lexicales de cette grammaire. Ensuite, pour chaque mot du lexique, nous sélectionnons à l'aide de son profil les classes qui le concernent. Nous obtenons des classes ancrées, c'est-à-dire des triplets (mot, profil morpho-syntaxique, description syntaxique) tels que :

– la description syntaxique est le contenu d'une classe lexicale sélectionnée par le mot ; cette description possède toujours un nœud syntaxique distingué comme ancre ;
– le profil résulte de l'unification du profil du mot et de celui de la classe lexicale sélectionnée.

En itérant cette opération pour toutes les entrées lexicales, nous transformons notre lexique descriptif en un lexique opérationnel pour l'analyse syntaxique. Voyons ce que peut être un exemple d'entrée d'un tel lexique opérationnel. Prenons le déterminant interrogatif *quelle*. Son profil nous est donné par le tableau 5.2. Parmi les 22 classes lexicales de la grammaire G_{wh} énumérées par le tableau 5.1, le profil du déterminant *quelle* en sélectionne 7 : *det.subj, det.no1.no2.verb-subj, det.no1.no2.subj-dupl, det.no1.no2.est-ce-que-sent, det.no1.yes2.verb-subj, det.no1.yes2.subj-verb, det.no1.yes2.est-ce-que-sent.*

A partir d'une entrée du lexique descriptif, nous obtenons 7 entrées du lexique opérationnel. Exhibons-en une particulière, celle qui lie le déterminant *quelle* à la classe *det.no1.no2.verb-subj*. Elle est formée par la classe ancrée de la figure 5.5. A l'aide de cette entrée, nous pourrons analyser les phrases suivantes :

à quelle personne pense Pierre ?

de quelle ville vient Pierre ?

à la mère de quelle personne pense Pierre ?

du sommet de quelle montagne part Pierre ?

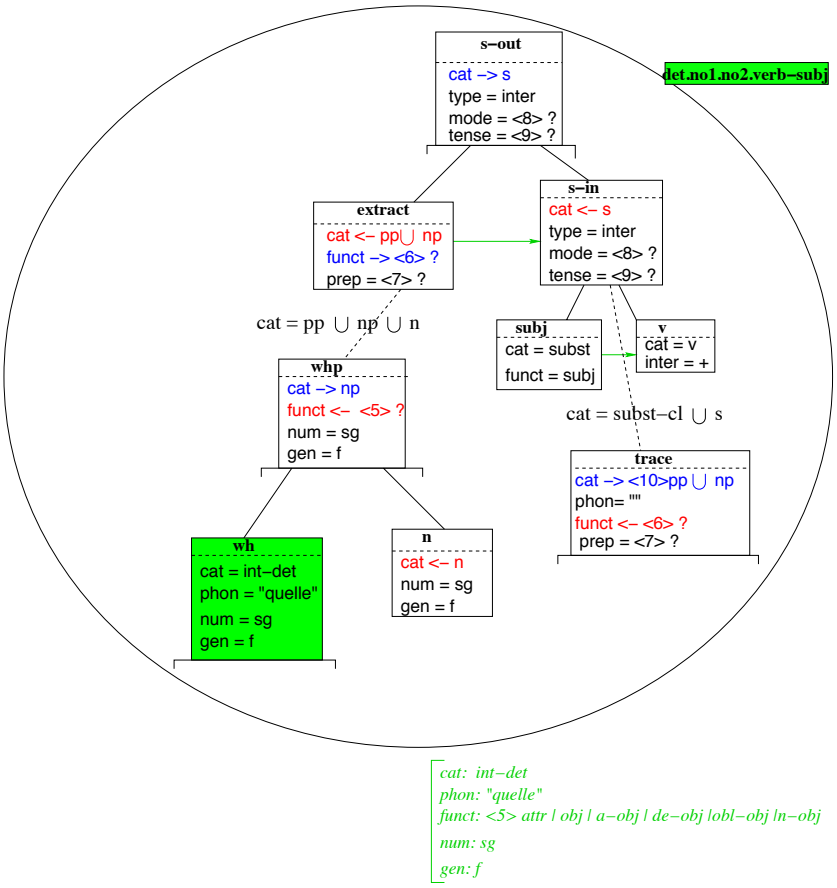


FIG. 5.5 – Contenu complet de la classe lexicale *det.no1.no2.verb-subj* ancrée par le mot *quelle*

En calculant d'abord tous les classes lexicales de la grammaire puis en les ancrant dans le lexique, on peut supposer que certains calculs ont été inutiles et on peut se demander s'il n'est pas plus judicieux de commencer par sélectionner à l'aide du profil d'un mot les classes de la grammaire qui sont pertinentes pour les croiser seulement après. Pour cela, il faut bien entendu parcourir la hiérarchie de haut en bas et le profil du mot va permettre d'élaguer celle-ci de toutes les branches inutiles. Si nous reprenons le mot *quelle* avec son profil, tel qu'il est donné par la table 5.2 et si nous appliquons cette méthode à la grammaire G_{wh}, nous obtenons la grammaire élaguée de la figure 5.6.

En conclusion, un certain nombre de grandes lignes ont été tracées pour une organisation modulaire et hiérarchisée des grammaires d'interaction. Certains choix sont encore à préciser comme celui de comment effectuer l'ancrage des classes terminales dans le lexique et celui qui est lié de comment effectuer le croisement des classes terminales. Pour avancer, il est nécessaire de passer à l'expérimentation mais pour cela il faut disposer de deux outils : un éditeur graphique de grammaires modulaires hiérarchisées et un générateur de lexiques opérationnels qui prenne en entrée une grammaire d'interaction hiérarchisée et un lexique descriptif. Ces deux outils restent à construire.

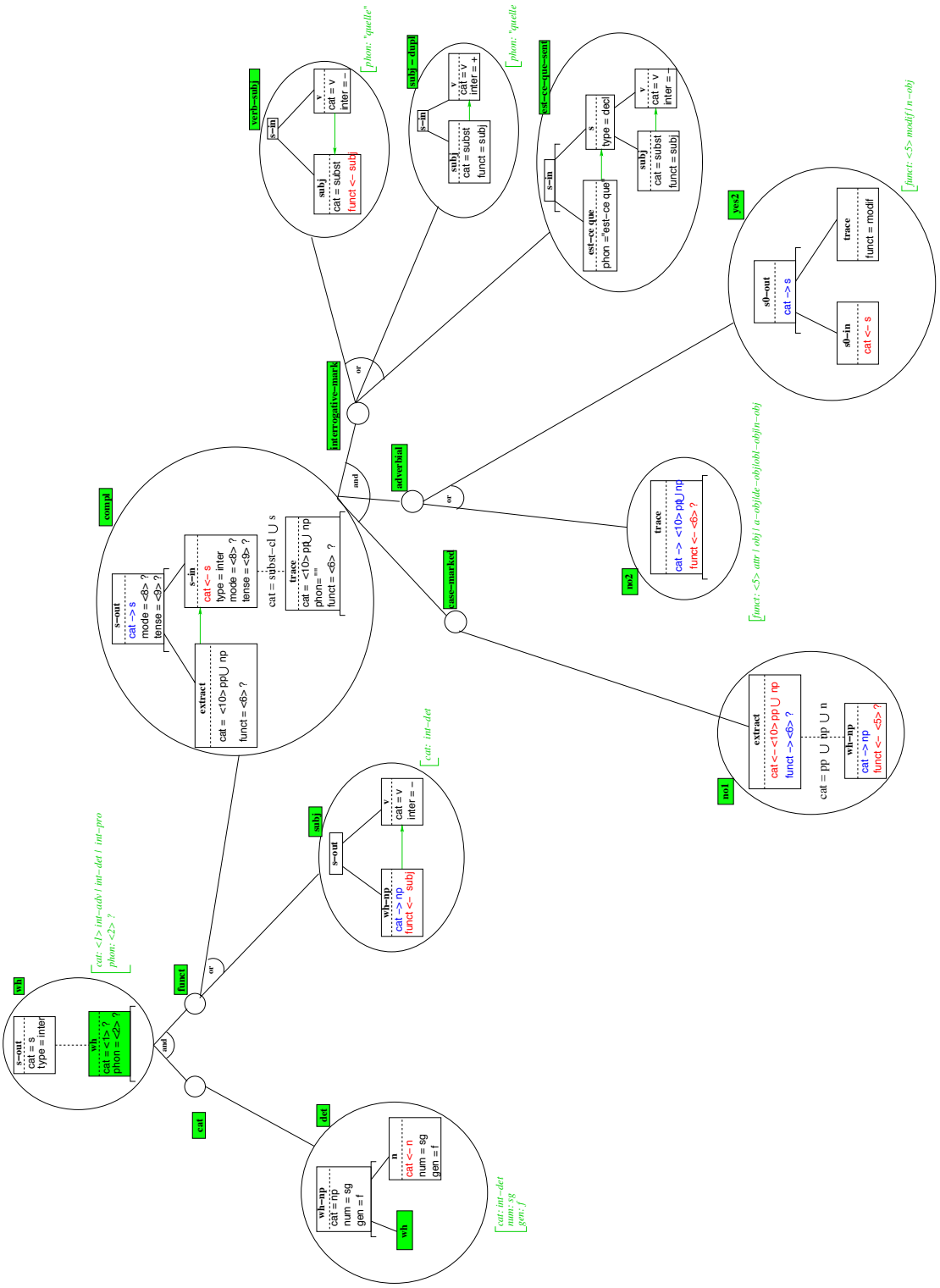


FIG. 5.6 – Grammaire d'interaction hiérarchisée extraite de G_{wh} pour le mot *quelle*

Chapitre 6

Comparaison avec d'autres formalismes et perspectives

Tout au long des chapitres précédents, nous avons déjà confronté les grammaires d'interaction avec d'autres formalismes grammaticaux sur des points particuliers. Nous voudrions maintenant reprendre plus systématiquement cette comparaison. Nous sommes bien conscients aussi que le terrain principal pour mener une telle comparaison devrait être le terrain expérimental, celui de l'application à des corpus réels, mais comme nous l'avons expliqué dans le chapitre précédent nous ne disposons pas de grammaires d'interaction à large couverture, nécessaires pour accomplir cette tâche. La comparaison que nous allons esquisser consistera avant tout à nous intéresser aux mécanismes et principes en œuvre dans les différents formalismes et à leur aptitude respective à capturer les propriétés des langues.

Nous ne discuterons pas de la complexité de l'analyse syntaxique relative aux différents formalismes. En effet, nous savons que le problème général de l'analyse syntaxique avec les grammaires d'interaction est NP-complet (voir l'introduction du chapitre 3). Mais nous avons vu dans le même chapitre que l'étiquetage électrostatique et l'analyse électrostatique contrôlée rendent le problème traitable ; cependant, l'évaluation théorique du gain en complexité qui en résulte reste à faire.

Nous commencerons par aborder des formalismes qui, si l'on regarde les principes en œuvre dans les grammaires d'interaction, présentent, en apparence du moins, des liens importants avec ces dernières :

- Les grammaires de propriétés de P. Blache [Bla01] s'appuient sur une vision d'une grammaire comme système de contraintes et il est clair que les grammaires d'interaction peuvent être vues de cette façon.
- Les grammaires minimalistes de E. Stabler [Sta97] fondent la composition syntaxique sur les opérations de fusion et de mouvement et toutes deux sont guidées par la neutralisation de traits polarisés.

Même si les rapports sont moins évidents, il était inévitable d'aborder les formalismes qui prévalent en linguistique informatique. Nous avons choisi trois paradigmes parmi les plus répandus dans le domaine : celui des TAG, celui d'HPSG et celui des grammaires de dépendance.

6.1 Les grammaires de propriétés

P. Blache [Bla01] présente les grammaires de propriétés comme découlant d'une nouvelle vision de l'analyse grammaticale fondée sur les contraintes. L'objectif est à la fois de mieux prendre en compte la souplesse des grammaires réelles et celle des énoncés qui ne sont pas toujours grammaticaux (notamment dans le dialogue oral). L'idée intéressante est de présenter une grammaire comme un système de contraintes, l'analyse consistant, pour un énoncé donné, à optimiser ces contraintes. Dans ces conditions, la réponse ne sera pas seulement de la forme oui/non mais une caractérisation de l'énoncé à l'aide des contraintes qu'il satisfait.

Si, dans les grammaires de propriétés, la forme des contraintes qui peuvent constituer une grammaire est décrite précisément, par contre il manque une description formelle de l'analyse syntaxique la ramenant à un problème mathématique de satisfaction ou d'optimisation de contraintes. Or, cette formalisation n'est pas triviale, notamment du fait que les grammaires de propriétés ne sont pas lexicalisées.

Les grammaires d'interaction se situent tout à fait dans la philosophie prônée par P. Blache mais la formalisation de l'analyse syntaxique comme un problème de satisfaction ou d'optimisation de contraintes y est aisée pour deux raisons :

- les grammaires sont lexicalisées ;
- les modèles des descriptions syntaxiques, les structures syntaxiques complètement spécifiées que nous avons appelées arbres syntaxiques, sont formalisées précisément.

L'analyse syntaxique peut être alors présentée comme un problème de satisfaction de contraintes où les variables sont :

- les positions relatives des nœuds syntaxiques des descriptions fournies par le lexique pour les mots de la phrase à analyser ;
- les valeurs des traits présents dans ces descriptions.

Les contraintes sur ces variables sont de deux types :

- les contraintes générales qui décrivent la forme des modèles, les arbres syntaxiques ici ; une contrainte de ce type sera par exemple d'exprimer qu'un nœud a au plus un père ;
- les contraintes spécifiques exprimant les descriptions syntaxiques fournies par le lexique pour les mots de la phrase, plus celles exprimant l'ordre de ces mots dans la phrase.

Si nous cherchons à résoudre ensuite toutes les contraintes, nous obtiendrons tous les arbres syntaxiques de la phrase analysée. Si nous nous contentons d'optimiser les contraintes, nous obtiendrons des structures syntaxiques partielles qui dépendront du critère d'optimisation choisi. Ce pourra être par exemple une forêt d'arbres partiels qui minimise le nombre de polarités non neutralisées. Ces considérations ne sont pas seulement théoriques : la première implémentation d'un analyseur syntaxique fondé sur les grammaires d'interaction s'est faite de cette façon en utilisant le langage de programmation avec contraintes *Oz* [Smo95]. Cette implémentation s'est aussi inspirée des travaux de D. Duchier et S. Thater [DT99] menés sur l'analyse syntaxique traduite comme un problème de satisfaction de contraintes pour un modèle très proche des gram-

maires d'interaction primitives définies à la section 1.5.

Ce problème de formalisation de l'analyse syntaxique en termes de de satisfaction de contraintes reste par contre un vide à combler pour les grammaires de propriétés.

Maintenant, si l'on compare le type de contraintes en œuvre dans les grammaires d'interaction avec celles en œuvre dans les grammaires de propriétés, on remarque que les premières mettent en jeu des arbres de profondeur pouvant être indéterminée alors que les secondes sont pour l'essentiel locales à un syntagme et ses constituants immédiats. Toutefois, dans ce domaine de localité réduit, elles permettent d'exprimer des propriétés que ne peuvent pas les grammaires d'interaction : on peut indiquer l'ordre de multiplicité d'apparition d'un type de sous-constituant et des nécessités de co-occurrence positives ou négatives entre sous-constituants. La représentation des dépendances non bornées à l'aide des grammaires de propriétés n'est pas vraiment réglée dans la mesure où les propriétés dynamiques destinées à les exprimer ne prennent en compte que des dépendances entre un syntagme et un constituant immédiat d'un autre syntagme qui se situe au même niveau que le premier. Le mécanisme qui permet la propagation au-delà n'est pas développé.

6.2 Les grammaires minimalistes

Les grammaires minimalistes de E. Stabler [Sta97] ne se placent pas sur le même plan que les grammaires d'interaction car elles se veulent une formalisation d'une théorie linguistique, le minimalisme de N. Chomsky [Cho95]. Les grammaires d'interaction ne sont liées à aucune théorie linguistique précise même si implicitement elles contiennent certains choix linguistiques.

Étant donné que la composition syntaxique et sémantique dans les grammaires minimalistes est fondée sur les deux opérations de fusion et de mouvement qui sont toutes deux guidées par la neutralisation de traits comme dans les grammaires d'interaction, on peut se demander s'il n'est pas possible de traduire les grammaires minimalistes sous forme de grammaires d'interaction particulières.

Une grammaire minimaliste est complètement lexicalisée et une entrée lexicale est une suite de traits syntaxiques, phonologiques et sémantiques. Les traits syntaxiques sont polarisés et sont divisés en trois catégories, selon qu'ils contrôlent la fusion, le mouvement ouvert, qui comprend les traits phonologiques et le mouvement caché, qui exclut les traits phonologiques. Le tableau 6.1 donne quelques exemples de ces traits.

opération contrôlée	traits positifs	traits négatifs
fusion	=c =d =n =t	c d n t v
mouvement ouvert	+CASE	-case
mouvement caché	+case	-case

TAB. 6.1 – Exemples de traits des grammaires minimalistes

Pour décrire les opérations de fusion et de mouvement, reprenons l'exemple

de grammaire utilisé dans [Sta97] en le simplifiant légèrement. Le lexique est donné par le tableau 6.2. Nous allons dériver un arbre syntaxique à partir de

=n d -case some
n student
=d +case =d v speaks
=n d -case every
n language
=v +CASE

TAB. 6.2 – Lexique définissant une grammaire minimaliste

ce lexique à l'aide des opérations de fusion et de mouvement. Le figure 6.1 donne la suite des pas de la dérivation. Certains ont été omis pour alléger la présentation.

Le pas 3 est un pas de fusion engendré par la neutralisation des traits $=d$ et d. L'élément porteur du trait positif $=d$ devient la tête du syntagme obtenu, ce qui est indiqué par le symbole $<$ qui est dirigé vers la tête.

Le pas 6 est un pas de mouvement caché engendré par la neutralisation des traits *+case* et *-case*. Le sous-arbre dont la feuille porteuse du trait *-case* est la tête va se dédoubler : un exemplaire avec comme seuls traits les traits phonologiques [25] va rester sur place et l'autre exemplaire avec les autres traits dont les traits sémantiques va devenir le spécifieur de l'arbre général dont la tête porte le trait *+case*.

Le pas 11 est un de mouvement ouvert engendré par la neutralisation des traits *+CASE* et *-case*. Le sous-arbre dont la feuille porteuse du trait *-case* est la tête va se déplacer complètement (avec les traits phonologiques cette fois) en position de spécifieur.

On obtient un arbre bien formé dans la mesure où tous les traits syntaxiques ont été effacés et on peut extraire de cet arbre une représentation sémantique sous forme arborescente et une représentation phonologique sous forme linéaire.

Dans les grammaires d'interaction, la neutralisation de traits entraîne la fusion de deux nœuds syntaxiques et, ce qui est fondamental, cette fusion ne rompt pas la monotonie de la description syntaxique dont elle est le siège : elle ne fait qu'ajouter de l'information sans en enlever.

Dans les grammaires minimalistes, la neutralisation de traits s'accompagne de modifications de structures plus complexes. Surtout, dans le cas de l'opération de mouvement, la monotonie est rompue. Même si on sait depuis les travaux de K. Vijay-Shanker sur les TAG [VS92] que l'on peut traduire une opération non monotone sous une forme monotone, la rupture de monotonie est ici suffisamment profonde pour que l'exercice consistant à traduire les grammaires minimalistes dans les grammaires d'interaction soit particulièrement difficile.

On retrouve ici l'opposition entre grammaires transformationnelles et grammaires déclaratives qui avait été un des facteurs expliquant l'introduction des

[25] Lorsque l'on écrit le trait *every*, c'est une abréviation pour la suite de traits */every/ (every)* dont le premier est la forme phonologique correspondant à la forme sémantique qui est donnée par le second.

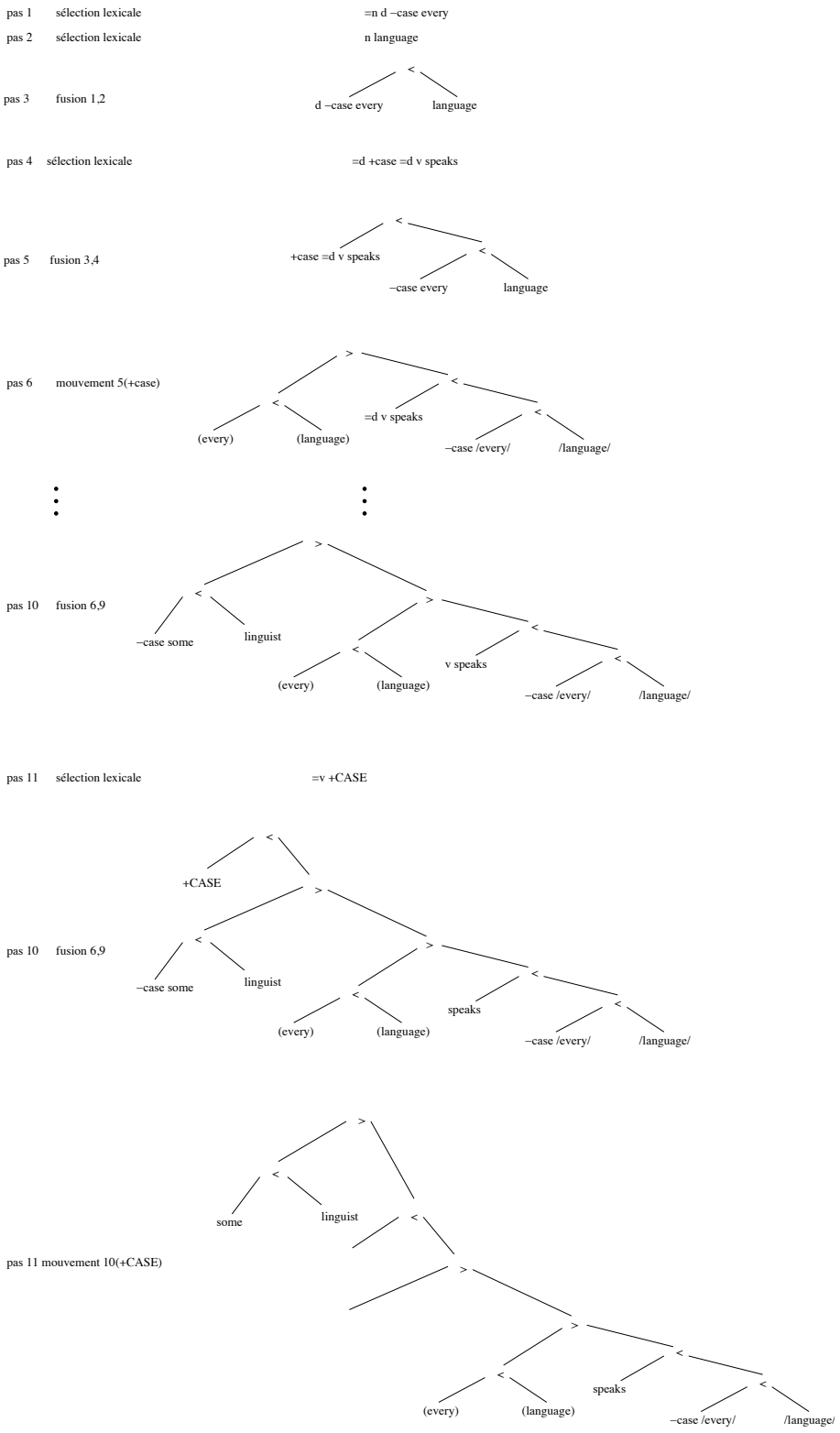


FIG. 6.1 – Dérivation d'un arbre syntaxique à partir d'une grammaire minimaliste

grammaires d'unification [Abe93] et il semble bien difficile de vouloir traduire une formalisation des premières dans un cadre prévu pour les secondes.

6.3 Les TAG

Les TAG [AR01] partagent avec les grammaires d'interaction le fait de pouvoir être présentées à l'aide des descriptions d'arbres. C'est K. Vijay-Shanker qui, le premier, a eu l'idée d'utiliser la notion de description d'arbres pour formaliser les TAG [VS92]. Son but était de rendre l'opération d'adjonction monotone. L'idée était très simple : elle consistait en placer dans tout arbre TAG les nœuds d'adjonction par des couples de quasi-nœuds liés par une relation de domination sous-spécifiée. En cas d'absence d'adjonction, les deux quasi-nœuds d'un même couple sont confondus, sinon ils s'identifient respectivement avec la racine et le pied d'un arbre auxiliaire. Les traits *top* et *bottom* des nœuds des arbres TAG se répartissent alors naturellement suivant les quasi-nœuds où cette distinction devient inutile. Le principe de l'analyse syntaxique peut alors être présenté comme la recherche de modèles de descriptions mais ces modèles sont soumis à des contraintes particulières telles que seule l'itération d'adjonctions permet de les obtenir [RVS94].

A partir de cette vision des TAG, une traduction dans les grammaires d'interaction vient naturellement à l'esprit. C'est d'ailleurs une traduction semblable qui est proposée par R. Muskens dans son formalisme LDG (Logical Description Grammar) [Mus01]. Elle consiste à polariser les arbres TAG de la façon suivante :

- pour tout couple de quasi-nœuds de catégorie grammaticale C, on affecte le trait $cat \leftarrow C$ et le trait $cat \rightarrow C$ respectivement au quasi-nœud supérieur et au quasi-nœud inférieur ;
- toute racine de catégorie grammaticale C est pourvue d'un trait $cat \rightarrow C$;
- tout nœud de substitution ou tout nœud pied de catégorie grammaticale C est pourvu d'un trait $cat \leftarrow C$.

Avec une telle traduction, il est facile de montrer que toute analyse TAG qui réussit peut être interprétée comme une analyse avec une grammaire d'interaction qui réussit.

Malheureusement, la réciproque n'est pas vraie. On peut trouver une description issue de la polarisation d'un ensemble d'arbres élémentaires TAG pour laquelle il existe un arbre syntaxique qui en est un modèle neutre et minimal mais qu'on ne puisse pas l'obtenir par une suite d'adjonctions et de substitutions. La raison en est que, dans les grammaires d'interaction, rien ne permet de contraindre à fusionner les deux quasi-nœuds d'un même couple avec la racine et le pied d'un même arbre auxiliaire. Il est facile de construire un contre-exemple qui montre le contraire. On ne peut donc pas plonger les TAG dans les grammaires d'interaction même si cela semble vrai pour les grammaires utilisées en pratique pour des langues comme le français ou l'anglais. Le fait que la traduction qui vient d'être présentée ne puisse pas être fondée théoriquement, n'enlève rien à son utilité pratique. Il peut être intéressant de polariser une grammaire TAG donnée pour lui appliquer ensuite les méthodes

d'analyse présentées au chapitre 3.

Même s'il est difficile d'établir théoriquement un rapport simple entre les TAG et les grammaires d'interaction du point de vue de leur pouvoir d'expression, les secondes ont un avantage sur les premières sur lequel nous avons déjà insisté au chapitre 2. La composition syntaxique y permet de superposer partiellement des arbres, ce qui donne une souplesse incomparable à l'écriture de grammaires lexicalisées. Nous ne sommes pas obligés de trancher dans le choix cornélien de savoir à quel mot rattacher telle ou telle construction grammaticale ; une même construction peut se rattacher partiellement à plusieurs mots en même temps, étant donné que l'on peut superposer des arbres syntaxiques. Cela évite par exemple de rattacher un nombre impressionnant de constructions au verbe. Les constructions interrogatives peuvent être attachées aux mots interrogatifs, les propositions relatives aux pronoms relatifs.

Il faut reconnaître que, malgré ces limites, il a été possible de construire et d'implémenter des TAG à large couverture pour l'anglais et pour le français. Pour l'anglais, c'est la grammaire intégrée au système XTAG [XTA01] et pour le français c'est la grammaire développée à Paris 7 par l'équipe d'A. Abeillé [Abe02]. Comme nous le soulignions dans l'introduction, ces travaux sont fondamentaux du point de vue de la démarche expérimentale qui est la nôtre.

Le point le plus faible des TAG est dans leur difficulté à s'interfacer avec un système de représentation sémantique. Nous avons déjà évoqué ce problème dans la section 4.6, où nous avons vu les limites des approches fondées sur les arbres de dérivation ou sur les TAG synchrones. L'approche de C. Gardent et L. Kallmeyer [GK03] qui vise à interfacer les arbres dérivés avec la *Hole Semantics* ouvre de nouvelles perspectives.

6.4 HPSG

HPSG [PS94] et les grammaires d'interaction partagent la même vision déclarative d'une grammaire comme système de contraintes. A partir de là, elles partagent la même vision de l'analyse syntaxique qui est de partir de descriptions structurelles sous-spécifiées et de les faire évoluer de façon monotone au cours du processus d'analyse vers des structures plus spécifiques.

Mais HPSG dispose d'une batterie de mécanismes qui en fait un formalisme linguistique extrêmement puissant du point de vue de son pouvoir d'expression :

- Les structures de traits et l'unification sont généralisées alors que dans les grammaires d'interaction les relations structurelles entre syntagmes sont traitées à part ; les structures de traits ne servent qu'à décrire les propriétés grammaticales des syntagmes mais en aucun cas leurs relations avec d'autres syntagmes.
- Les structures de traits d'HPSG sont récursives en ce sens que les valeurs des traits peuvent être elles-mêmes des structures de traits, alors que, dans les grammaires d'interaction, les valeurs des traits sont des disjonctions d'atomes. Combiné à la ré-entrance des traits, cela permet à HPSG de transmettre les traits de façon élégante et économique. Pre-

nons par exemple le principe de tête d'HPSG. Un syntagme S et sa tête T partage un trait $HEAD$ dont la valeur est une structure de traits. Si, au cours du processus de composition syntaxique, un trait $NUM = sg$ vient s'ajouter à la valeur du trait $HEAD$ de T, il sera automatiquement ajouté à la valeur du trait $HEAD$ de S. Avec les grammaires d'interaction, ce n'est pas possible. Il faut prévoir au départ tous les traits qui pourront être partagés, ce qui est relativement lourd. Rien n'empêche cependant d'étendre les grammaires d'interaction pour permettre, de façon limitée peut-être, que certaines valeurs de traits soient elles-mêmes des structures de traits.

- Le partage de valeurs entre traits est largement utilisé à travers la réentrance de valeurs. C. Pollard et I. Sag le disent eux-mêmes [PS94] : ce mécanisme est au centre d'HPSG quant à la réalisation des différentes formes de dépendance. Les structures syntaxiques sont parfois complexes ; contrairement aux grammaires d'interaction qui se restreignent à des arbres, ce sont des DAG, ce qui permet par exemple d'exprimer des phénomènes tels que la redistribution des compléments d'un verbe par un verbe dit restructurant. Dans la section 2.3.3, nous en avons eu une illustration avec les verbes causatifs.
- Enfin, le typage des structures de traits d'HPSG est un moyen supplémentaire d'exprimer des contraintes de façon économique.

Si les TAG ne permettent pas d'effectuer de la superposition d'arbres, l'unification le permet dans HPSG, étant donné que la structure syntagmatique est codée dans les structures de traits.

On vient de voir des différences entre HPSG et les grammaires d'interaction qui font apparaître la puissance d'expression supérieure du premier formalisme. Il est maintenant d'autres différences qui ne sont pas forcément à l'avantage d'HPSG :

- Dans HPSG, le principe de composition de base est le même que celui qui est appliqué pour les grammaires algébriques : des règles grammaticales permettent de composer les syntagmes. Ces règles sont cependant plus générales et plus puissantes que celles qui définissent des grammaires algébriques. Les constructions grammaticales qui ne sont pas canoniques font appel à des règles de transformation lexicales. Il y a donc un partage des rôles entre le lexique et les règles grammaticales que l'on ne trouve pas dans les grammaires d'interaction qui sont entièrement lexicalisées ; ces dernières n'ont qu'une seule règle de composition, la neutralisation de traits. Ce mécanisme qui est au centre des grammaires d'interaction se retrouve dans HPSG comme principe particulier, le principe de sous-catégorisation.
- Pour modéliser les dépendances non bornées dans les grammaires d'interaction, on utilise une relation de domination sous-spécifiée mais contrainte qui respecte le principe de localité étendue. HPSG rompt avec ce principe en utilisant des traits non locaux qui sont propagés de proche en proche vers le haut depuis la tête lexicale qui leur a donné naissance jusqu'au syntagme où ils sont capturés. Comme nous l'avons déjà dit, l'expression des dépendances non bornées dans les grammaires d'interaction est plus proche du concept d'incertitude fonctionnelle utilisé dans LFG [KZ89].

Pour ce qui est de l'intégration de la sémantique, le trait SEM associé à chaque syntagme contient dans sa valeur un trait $RESTR$ qui une suite de relations prédicat-arguments et qui sont un codage du DAG sémantique des grammaires d'interaction. Dans HPSG, ce DAG est complètement spécifié et pour exprimer que tel individu ou tel prédicat est dans le champ de tel autre prédicat, cela ne peut pas se faire directement mais par le biais de traits supplémentaires tels que SIT qui indique la situation ou l'événement dans lequel intervient un individu ou un prédicat donné. L'interface syntaxe-sémantique s'effectue de la même façon qu'avec les grammaires d'interaction par un trait $INDEX$ qui, donne pour chaque syntagme, le nom de l'individu ou de la situation qu'il représente.

En conclusion, il est clair que toute la machinerie dont dispose HPSG lui donne un pouvoir d'expression supérieur aux grammaires d'interaction mais la question qui vient en corollaire, c'est :"A quel prix ?" D'un point vue informatique, il est clair que l'écriture d'algorithmes d'analyse performants en est rendue d'autant plus difficile.

6.5 Les grammaires de dépendances

Contrairement aux TAG qui sont un formalisme linguistique bien précis, les grammaires de dépendance constituent plutôt un paradigme qui s'exprime à travers différentes théories et formalismes [Mel88, Hud90].

A la différence des TAG et des grammaires d'interaction, ce n'est pas la notion de syntagme qui est la notion de base mais celle de dépendance entre mots. Mais cette notion de dépendance a beaucoup de parenté avec celle de polarité qui est au centre des grammaires d'interaction mais aussi des grammaires catégorielles. Cela est d'ailleurs mis en évidence par l'équivalence qui a été établie entre les grammaires AB de Bar-Hillel et les grammaires de dépendance de Hays [Gai65]. Dans une dépendance d d'un mot w_1 par rapport à un mot w_2, on peut dire que w_2 est porteur d'une polarité $-d$ alors que w_1 est porteur d'une polarité $+d$ et que la dépendance va se réaliser dans une neutralisation de ces polarités. Cette remarque est très générale mais voyons à travers un exemple précis comment elle peut permettre de présenter une grammaire de dépendance comme une grammaire d'interaction.

Considérons une présentation formelle lexicalisée des grammaires de dépendance, choisissons celle de D. Duchier [DD01]. Elle comprend deux niveaux, un niveau syntaxique et un niveau topologique. Nous ne nous intéresserons qu'au niveau syntaxique. Une entrée lexicale d'un mot w comporte deux traits : un trait *cat* qui donne l'ensemble des catégories grammaticales possibles du mot w et un trait *valency* qui donnent l'ensemble des dépendances syntaxiques qui sont issues de w. Par exemple, pour le verbe *versucht*, le lexique nous donne :

$$cat = \{vfin\} \quad valency = \{subject,\ zuinf\}$$

Cela signifie que le verbe *versucht* a une seule catégorie, celle d'un verbe conjugué, et il gouverne deux dépendances, une dépendance sujet et une autre complément à l'infinitif introduit par *zu*.

Les dépendances sont accompagnées de contraintes concernant la catégorie grammaticale du dépendant et l'accord entre le gouverneur et le dépendant.

Dans notre exemple, ces contraintes indiquent d'abord que le sujet doit avoir la catégorie *n* (nom) et que les traits d'accord du verbe et du sujet doivent être les mêmes. Pour l'autre dépendance, la seule contrainte est que le dépendant doit avoir la catégorie *zuinf* d'un verbe à l'infinitif.

Il est très facile de traduire une telle entrée lexicale avec les contraintes qui l'accompagnent par une description syntaxique d'une grammaire d'interaction. La figure 6.2 nous montre ce que pourrait être une telle description. Étant donné la présentation syntagmatique des grammaires d'interaction, il est

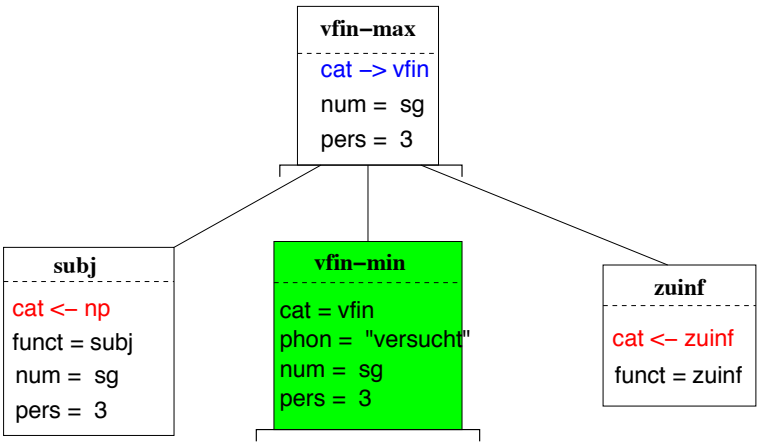


FIG. 6.2 – Description syntaxique traduisant l'entrée lexicale du verbe *versucht* pour une grammaire de dépendance

nécessaire d'ajouter au nœud *vfin-min* qui représente le mot *versucht* un nœud *vfin-max* qui représente sa projection selon la terminologie des grammaires de dépendance, c'est-à-dire le mot avec tous ses dépendants directs et indirects.

Il y a un seul obstacle à la traduction des grammaires de D. Duchier sous forme de grammaires d'interaction, c'est que la notion de dépendance y est plus fine que celle de polarité : une dépendance peut être optionnelle ou itérée ou les deux. Une dépendance optionnelle peut correspondre par exemple à un complément optionnel alors qu'une dépendance optionnelle itérée peut correspondre à une modification d'un nom par un adjectif. Ceci peut d'ailleurs suggérer un raffinement des polarités des grammaires d'interaction pour se rapprocher de ces différents types de dépendances.

Le lien entre grammaires de dépendance et grammaires d'interaction apparaît encore plus clairement dans le formalisme développé par A. Dikovsky [Dik01]. Celui-ci se situe dans la cadre des grammaires de dépendance dont le noyau est exprimé par un ensemble de règles non contextuelles. Les dépendances locales sont exprimées directement entre les éléments des parties droite des règles grammaticales et, c'est là l'originalité du modèle, les dépendances non bornées utilisent la polarisation de non-terminaux. La réalisation de ces dépendances va s'effectuer au cours de l'analyse par la neutralisation de polarités comme dans les grammaires d'interaction. Cependant, le fait que ce mécanisme soit utilisé dans un cadre plus restreint et plus contraint, permet de ramener l'analyse dans le cadre de grammaires algébriques, ce qui est intéressant du

point de vue complexité.

Venons en à une différence importante entre grammaires de dépendance et grammaires d'interaction. Les nœuds des arbres de dépendances sont les mots de la langue alors que ceux des descriptions syntaxiques des grammaires d'interaction sont des syntagmes qui peuvent même être vides. D'un point de vue computationnel, les premiers présentent un avantage : pour une phrase donnée de n mots, on connaît les nœuds de l'arbre de dépendance qui sont les n mots. Par contre, cela peut être un inconvénient pour représenter certaines interactions complexes entre mots comme par exemple la dépendance non bornée en cascade dans les relatives 2.3. Il est alors nécessaire d'imaginer des mécanismes spécifiques pour exprimer ces interactions tels que les arbres à bulles [Kah00].

Pour ce qui est de l'intégration de la sémantique, les grammaires de dépendance n'offrent pas une seule approche mais celle qui est développée dans la théorie Sens-Texte de I. Melcuk [Mel88] et qui a été formalisée par S. Kahane [Kah02] est très voisine de celle qui est mise en œuvre dans les grammaires d'interaction synchrones. Nous en avons déjà parlé dans la section 4.5 donc nous n'y reviendrons pas.

6.6 Perspectives

En présentant les grammaires d'interaction comme un nouveau formalisme linguistique, nous souhaitons rester modestes. Nous savons très bien que la modélisation des langues est une tâche extrêmement difficile et que LE formalisme qui viendra tout résoudre n'existe pas. Convaincus que l'approche symbolique a encore de beaux jours devant elle, nous pensons que l'essentiel en la matière est d'avoir une démarche scientifique. Nous avons donné un certain nombre d'arguments théoriques en faveur des grammaires d'interaction mais seule l'expérimentation à travers la confrontation avec des corpus réels permettra d'évaluer vraiment la pertinence du formalisme. Il permettra sûrement aussi de le faire évoluer. Par exemple, il pourrait être intéressant de raffiner le système de polarités pour aller dans le sens de ce qui se fait avec les grammaires de dépendances. L'utilisation de structures de traits comme valeurs de traits, comme le fait HPSG mais de façon plus limitée, devrait contribuer à éviter certaines lourdeurs.

Pour aller vers l'expérimentation, il est un chantier incontournable qu'il est nécessaire d'aborder : c'est celui de la construction de grammaires à large couverture. Cela pose à la fois des questions linguistiques et informatiques qu'il s'agit de résoudre et cela nécessite de créer les logiciels informatiques qui permettront la construction de telles grammaires. Ce sera une de nos priorités pour l'avenir.

Bibliographie

[Abe91] A. Abeillé. *Une grammaire lexicalisée d'Arbres Adjoints pour le français.* Thèse de doctorat, Université Paris 7, 1991.

[Abe93] A. Abeillé. *Les nouvelles syntaxes.* Linguistique. Armand Colin, 1993.

[Abe02] A. Abeillé. *Une grammaire électronique du français.* CNRS Editions, Paris, 2002.

[Adj35] K. Adjukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1 :1–27, 1935.

[AGMS98] A. Abeillé, D. Godard, P. Miller, and I. Sag. French bounded dependencies. In S. Balari and L Dini, editors, *Romance in HPSG.* CSLI Publications, Standford, 1998.

[AR01] A. Abeillé and O. Rambow, editors. *Tree adjoining grammars : formalism, linguistic analysis and processing.* Stanford, CSLI, 2001.

[BBL+96] J. Bos, G. Björn, C. Lieske, Y. Mori, M. Pinkal, and K. Worm. Compositional semantics in verbmobil. In *15th International Conference on Computational Linguistics, COLING'96, Copenhagen, Denmark*, 1996.

[BdG01] Guillaume Bonfante and Philippe de Groote. Stochastic Lambek Categorial Grammars. In Lawrence Moss, editor, *Formal Grammars and Mathematics of Langage Conference - FGMOL'01, Helsinki, Finland*, number 5 in Electronic Notes in Computer Science, 2001.

[Bla01] P Blache. *Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles.* Hermès Sciences, 2001.

[Bos95] J Bos. Predicate logic unplugged. In P. Dekker and M. Stokhof, editors, *10th Amsterdam Colloquium*, pages 133–142, 1995.

[Bou03] P. Boullier. Supertagging : a Non-Statistical Parsing-Based Approach. In *8th International Workshop on Parsing Technologies (IWPT'03), Nancy, France, 2003*, pages 55–66, 2003.

[Can99] M.-H Candito. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien.* Thèse de doctorat, Université Paris 7, 1999.

[Car92] B. Carpenter. *The logic of typed feature structures.* Cambridge University Press, Cambridge, 1992.

[Car98] B. Carpenter. *Type-logical Semantics.* MIT Press, Cambridge, Massachusetts, 1998.

[CFS99] A. Copestake, D. Flickinger, and I.A. Sag. Minimal Recursion Semantics - an Introduction. Draft, 1999.

[Cho57] N. Chomsky. *Syntactic Structures.* The Hague : Mouton, 1957.

[Cho95] N. Chomsky. *The Minimalist Program.* MIT Press, Cambridge, Massachusetts, 1995.

[CNS+98] J. Carroll, N. Nicolov, O. Shaumyan, M. Smets, and D. Weir. The LEXSYS Project. In *4th International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 29–33, 1998.

[DD01] D. Duchier and R. Debusmann. Topological dependency trees : A constraint-based account of linear precedence. In *39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France, 2001.

[dG99] P. de Groote. An algebraic correctness criterion for intuitionistic multiplicative proofnets. *Theoretical Computer Science*, 224 :115–134, 1999.

[Dik01] A. Dikovsky. Grammars for Local and Long Dependencies. In *Meeting of the Association for Computational Linguistics*, pages 156–163, 2001.

[DT99] D. Duchier and S. Thater. Parsing with tree descriptions : a constraint based approach. In *Natural Language Understanding and Logic Programming NLULP'99,Dec 1999, Las Cruces, New Mexico*, 1999.

[EG96] R. Evans and G. Gazdar. DATR : a Language for Lexical Knowledge Representation. *Computational Linguistics*, 22(2) :167–247, 1996.

[EGW95] R. Evans, G. Gazdar, and D. Weir. Encoding Lexicalized Tree Adjoining Grammars with a nonmonotonic inheritance hierarchy. In *33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 77–84, 1995.

[ENRX98] M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over lambda structures in semantic underspecification. In *17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'98), Montreal,Canada, 1998*, 1998.

[Fal01] Y. Falk. *Lexical-Functional Grammar : an introduction to parallel constraint-based syntax.* Standford : Center for the Study of Language and Information, 2001.

[Gai65] H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 18 :304–337, 1965.

[GCR02] B. Gaiffe, B. Crabbé, and A. Roussanaly. Metagrammar Compiler. In *6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6), Venice, Italy*, 2002.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1) :1–102, 1987.

[GK03] C. Gardent and L. Kallmeyer. Semantic construction in FTAG. In *EACL'2003, Budapest, Hungary*, 2003.

[Hud90] R. Hudson. *Recent Trends in Dependency Syntax*, pages 329–338. Handbücher zur Sprach- und Kommunikationswissenschaften. de Gruyter, 1990. Band 9 - Syntax.

[JK97] A. K. Joshi and S. Kulick. Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*, 20(6) :637–667, 1997.

[Joh98] M. Johnson. Proof Nets and the Complexity of Processing Center-Embedded Constructions. *JOLLI*, 7(4) :433–447, 1998. Special Issue of the Workshop on Logical Aspects of Computational Linguistics, Nancy, september 1995.

[JS94] A. Joshi and B. Srinivas. Disambiguation of super parts of speech (or supertags) : Almost parsing. In *COLING'94, Kyoto*, 1994.

[Kah97] S. Kahane. Bubble trees and syntactic representations. In T. Becker and U. Krieger, editors, *5th Meeting of the Mathematics of Language (MOL5), Saarbrücken*, pages 70–76, 1997.

[Kah00] S. Kahane. Extractions dans une grammaire de dépendance à bulles. *T.A.L.*, 41(1) :187–216, 2000.

[Kah02] S. Kahane. *Grammaire d'Unification Sens-Texte - Vers un modèle mathématique articulé de la langue*. Habilitation à diriger des recherches, Université Paris 7, 2002.

[Kal99] L. Kallmeyer. *Tree Description Grammars and Underspecified Representations*. PhD thesis, Universität Tübingen, 1999.

[Kan92] M. Kanovich. Horn programming in linear logic is NP-complete. In *Seventh Annual Symposium on Logic in Computer Science, Santa Cruz, California*, pages 200–210. IEEE Computer Society Press, Los Alamitos, California, 1992.

[KCdK00] S. Kahane, M.-H. Candito, and Y. de Kercadio. An alternative description of extractions in TAG. In *Workshop TAG+5, Paris*, pages 115–122, 2000.

[KJ85] A. Kroch and A. Joshi. The Linguistic Relevance of Tree Adjoining Grammars. Technical Report MS-CI-85-18, Department of Computer and Information Science, University of Pennsylvania, 1985.

[KNT01] A. Koller, J. Niehren, and R. Treinen. Dominance constraints : Algorithms and complexity. In M. Moortgat, editor, *Third International Conference on Logical Aspects of Computational Linguistics (LACL'98), Grenoble, France,1998*, volume 2014 of *Lecture Notes in Artificial Intelligence*, 2001.

[Kro87] A. Kroch. Subjacency in a Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language (MOL1.* John Benjamins, 1987.

[KZ89] R. Kaplan and A. Zaenen. Long-distance dependencies, constituent structure and functional uncertainty. In *Alternative Conceptions of Phrase Structure*, pages 17–42. Chicago University Press, Chicago, 1989.

[Lam58] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65 :154–169, 1958.

[Lam61] J. Lambek. On the calculus of syntactic types. In R. Jakobson, editor, *Structure of Language and its Mathematical Aspects, 12th Symposium in Applied Mathematics, Providence, Rhode Island, april 1960*, pages 166–178. American Mathematical Society, 1961.

[Lam96] F. Lamarche. From proof nets to games. *Electronic Notes in Theoretical Computer Science*, 3, 1996. Special Issue of Linear Logic'96, Tokyo Meeting, march 1996.

[LORP+96] Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996,Kopenhagen*, 1996.

[LR98] A Lecomte and C. Retoré. Words as Modules : a Lexicalised Grammar in the framework of Linear Logic Proof Nets. In C. Martin-Vide, editor, *Mathematical and Computational Analysis of Natural Language*. John Benjamins, 1998.

[Mel88] I. Mel'cuk. *Dependency Syntax : Theory and Practice*. Albany, N.Y. : The SUNY Press, 1988.

[MHF83] M. Marcus, D. Hindle, and M. Fleck. D-Theory : Talking about Talking about Trees. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 129–136, 1983.

[Moo88] M. Moortgart. *Categorial Investigations : Logical and Linguistic Aspects of the Lambek Calculus*. Foris, Dordrecht, 1988.

[Moo96] M. Moortgart. Categorial Type Logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier, 1996.

[Mor94] G. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

[Mor00] G. Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3) :319 – 338, 2000.

[MS97] P. Miller and I Sag. French Clitic Movement without Clitics or Movement. *Natural Language and Linguistic Theory*, 15(3) :573–639, 1997.

[Mus01] R. Muskens. Talking about trees and truth-conditions. *Journal of Logic, Language, and Information,*, 10(4) :417–455, 2001.

[Per99] G. Perrier. Labelled proof nets for the syntax and semantics of natural languages. *Logic Journal of the IGPL*, 7(5) :629–654, September 1999. Registred as a LORIA publication under 99-R-139.

[Per01] G. Perrier. Intuitionistic proof nets as models of directed acyclic graph descriptions. In R. Nieuwenhuis and A. Voronkov, editors, *8th International Conference LPAR 2001, Havana, Cuba, 2001*, volume 2250 of *Lecture Notes in Computer Science*, pages 233–248. Springer Verlag, 2001.

[PS94] Carl J. Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[Ret00] C. Retoré. The Logic of Categorial Grammars, 2000. avalaible at URL : http ://www.cs.bham.ac.uk/research/conferences/esslli/.

[Rey93] U. Reyle. Dealing with ambiguities by underspecification : Constrcution, Representation and Deduction. *Journal of Semantics*, 10 :123–179, 1993.

[Roo91] D. Roorda. *Resource Logics. Proof-Theoritical Investigations*. PhD thesis, Universiteit van Amsterdam, 1991.

[RS96] O. Rambow and G. Satta. Synchronous Models of Language. In *ACL'96, Santa Cruz*, 1996.

[RVS92] J. Rogers and K. Vijay-Shanker. Reasoning with descriptions of trees. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 72–80, 1992.

[RVS94] J. Rogers and K. Vijay-Shanker. Obtaining trees from their descriptions : an application to tree-adjoining grammars. *Computational Intelligence*, 10(4) :401–421, 1994.

[RVSW95] O. Rambow, K. Vijay-Shanker, and D. Weir. D-Tree Grammars. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158, 1995.

[RVSW01] O. Rambow, K. Vijay-Shanker, and D. Weir. D-tree substitution grammars. *Computational Linguistics*, 27(1) :87–121, 2001.

[Shi94] S Shieber. Restricting the Weak-Generative Capacity of Synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 10(4) :371–385, 1994.

[Smo95] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.

[SS90] S. Shieber and Y. Schabes. Synchronous Tree-Adjoining Grammars. In *3th International Conference on Computational Linguistics*, volume 3, pages 1–6, 1990.

[Sta97] E. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics, LACL'96, Nancy, France, September 1996*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, 1997.

[Ste00] M. Steedman. *The Syntactic Process*. Bradford Books. MIT Press, 2000.

[Tes59] L. Tesnière. *Eléments de syntaxe structurale*. Librairie C. Klincksieck, Paris, 1959.

[vDP96] K. van Deemter and S. Peters, editors. *Semantic Ambiguity and Underspecification*. CSLI, Standford, 1996.

[VS87] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, 1987.

[VS92] K. Vijay-Shanker. Using description of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4) :481–517, 1992.

[Wah93] W. Wahlster. Verbmobil : Translation of face-to-face dialogs. In *3th European Conference on Speech Communication and technology, Berlin*, pages 29–38, 1993.

[XTA01] Research Group XTAG. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania, 2001.